BEGA220A
USER MANUAL

Features:

- **ARM 9 CPU**
- **7” 800X480 TFT with Touch Panel**
- **Wi-Fi**
- **10/100Mbps Ethernet**
- **RS-232/485/422**
- **Audio**
- **SD/MMC**
- **WINCE OS**



BOLYMIN, INC.
13F-1, No.20, Ta-Long Rd., 40310 Taichung, Taiwan, R.O.C.
WEB SITE:http://www.bolymin.com.tw   TEL:+886-4-23293029 FAX:+886-4-23293055

History of Version

Version	Contents	Date	Note
01	NEW VERSION	2010/05/17	SPEC.
02	Revision – evc4sp4 , driver download URL,etc.	2010/10/18	SPEC

TABLE OF CONTENTS

1 General Information

This chapter provides basic information about Bolymin's BEGA220A module and it consists of :

1.1 Introduction
1.2 Specifications
1.3 Mechanical Specifications

1.1 Introduction

BEGA220A module is a general purpose embedded system and is suitable for versatile applications such as medical probing devices, in-car automation, human machine interface (HMI), etc. And here is the order information for the BEGA220A family :

Order Information

Part No.	RS-485	RS-422	WLAN	20 PIN EXT BUS(**IOX12,ADCX6**)
BEGA220A	☆			
BEGA220A1	☆		☆	
BEGA220A2	☆			☆
BEGA220A3	☆		☆	☆
BEGA220A4		☆		
BEGA220A5		☆	☆	
BEGA220A6		☆		☆
BEGA220A7		☆	☆	☆

1.1.1 Packing Content

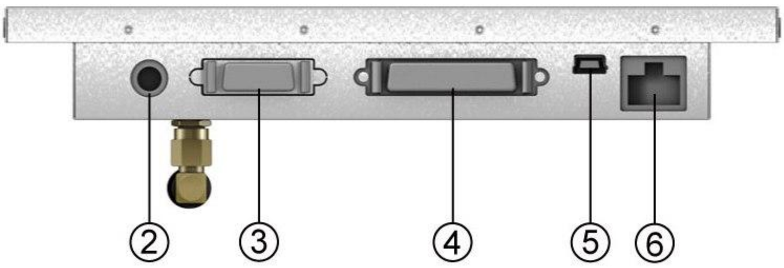
A complete package should contain all of the following:

- BEGA220A module
- Female USB to mini USB cable with 20cm length (CB04P201LC01$)
- 220 communication cable with 3M 40 pin connector and length100cm(CB40P1000LC01$)(only on sample stage)
- CD for user manual and utility software

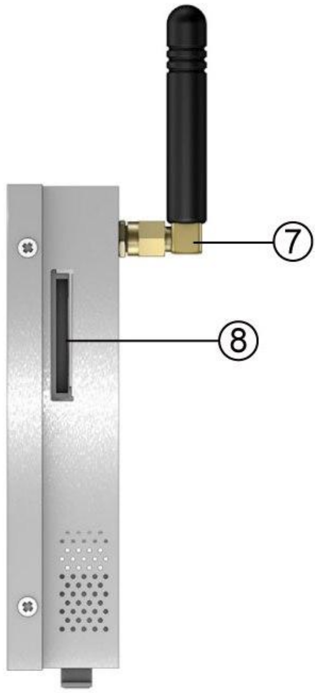
1.1.2 Module Layout



No.	Name	Description
1	Touch screen	To support touch screen operation on BEGA220A



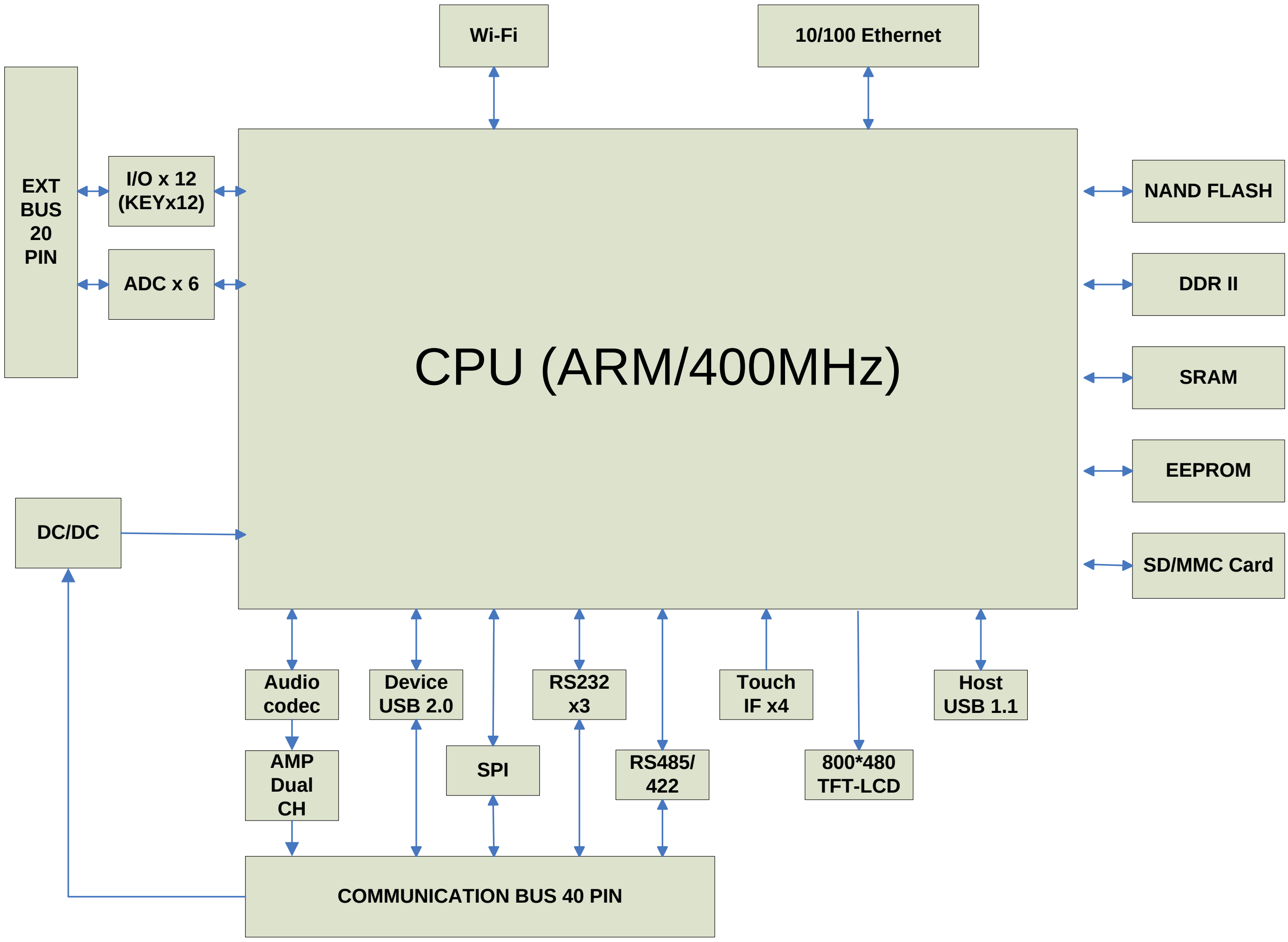
No.	Name	Description
2	Power socket	Use 12V/3A power with a spec. of DC Plug_in φ 2.5 socket
3	ADC_GPIO Bus	GPIOx12, ADCx6。
4	Communication Bus	SPI x1，Device USB 2.0，RS-232x3，RS485/422x1(Optional)。
5	Host-USB 1.1	Connect to USB 1.1 peripheral (keyboard, mouse,etc)
6	RJ-45	Connect to Ethernet network



No.	Name	Description
7	Wi-Fi Antenna socket	Connect Wi-Fi antenna
8	SD Card socket	Connect SD and MMC Card (4GB max.)

1.2 Specifications

1.2.1 System Functional Blocks

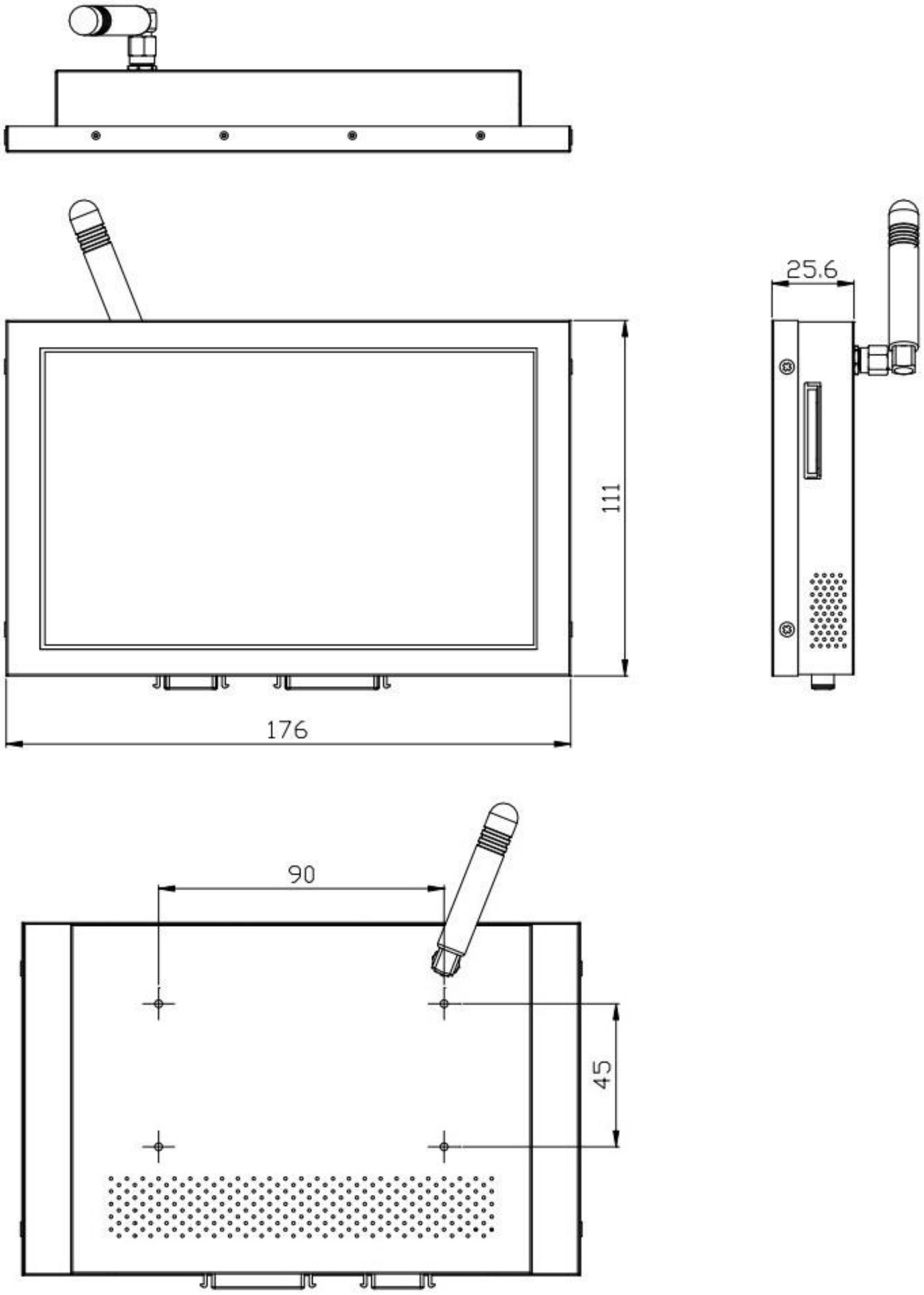


1.2.2 Module Specifications

Parameter	Specification
CPU	◆ Samsung S3C2416X 400MHz ◆ 32 bit RISC architecture ARM926EJ CPU core
System Memory	◆ 16-bit 64MB/133MHz DDR2 memory
Storage Device	◆ 2GB NAND Flash
Series Port	◆ 2 wire RS-232 x 3 ◆ Isolated RS485/422 x1(Option) ◆ SPI x1(Option)
USB	◆ 1x USB device (USB2.0) for Active Sync only ◆ 1XUSB host (USB1.1)
GPIO	Support programmable 12 x IO sharing with Key board interface
ADC	Support 6 x channel 12 bit high speed A/D converter
LAN	High performance 16-bit 10/100 Ethernet controller
Audio	Dual channels 2 watts speaker output
Wi-Fi	IEEE 802.11b/g,Wi-Fi compliant
OS	◆ WinCE 5.0(default)
LCD Size	7” TFT LCD
LCD Resolution	800x480
LCD Brightness	400 cd/m^2
Power Supply	DC9V~DC28V
Operating Temperature	-20℃ ~ +70℃

1.3 Mechanical Specifications

1.3.1 Module Specifications



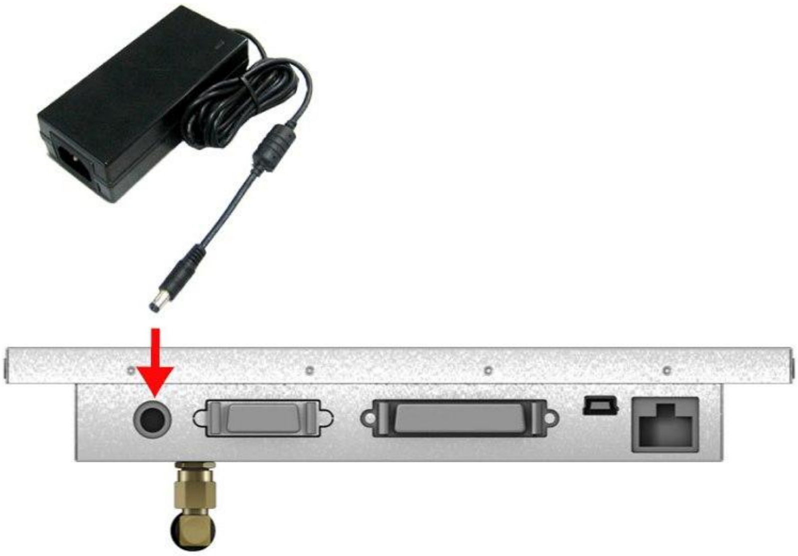
2 BEGA220A Installation & Testing

This chapter provides installation information for BEGA220A module and it consists of :

2.1 Connect power
2.2 Connect SD/MMC Card
2.3 Connect Host USB
2.4 Connect LAN
2.5 Connect Wi-Fi
2.6 Connect Communication Bus
2.7 Connect ADC_GPIO Bus

2.1 Connect Power

User may prepare a power adaptor with an output of DC12V/3A and a 2.5φ as illustrated.



DC Plug_in φ 2.5

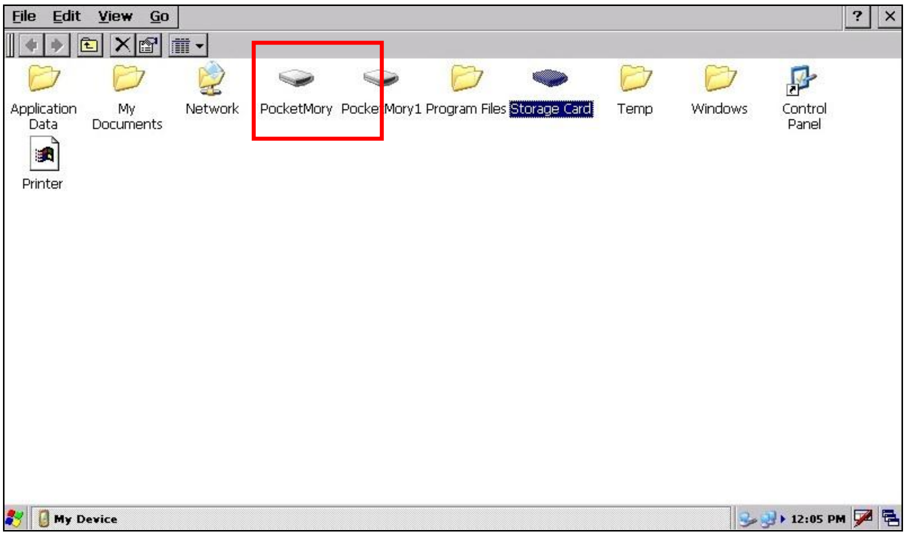


2.2 Connect a SD/MMC Card

Insert SD/MMC card as illustrated and eject card only when it's not at reading mode. Note that there is write-protection toggle switch on the card and make sure it's not write-protected so data can be written into the card.

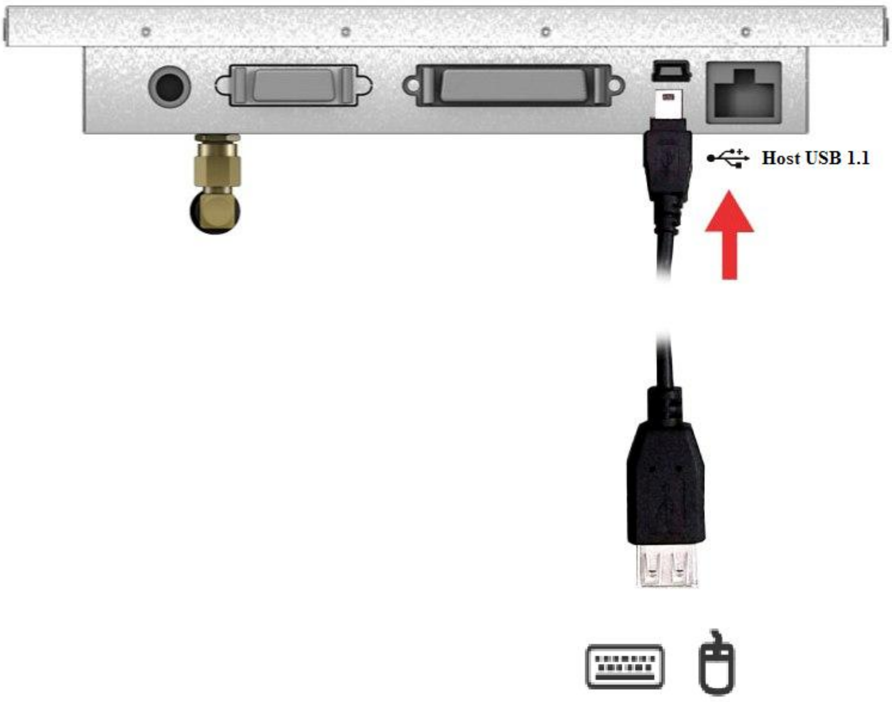


WinCE will then detect the SD/MMC Card and appears a storage card icon as follows:



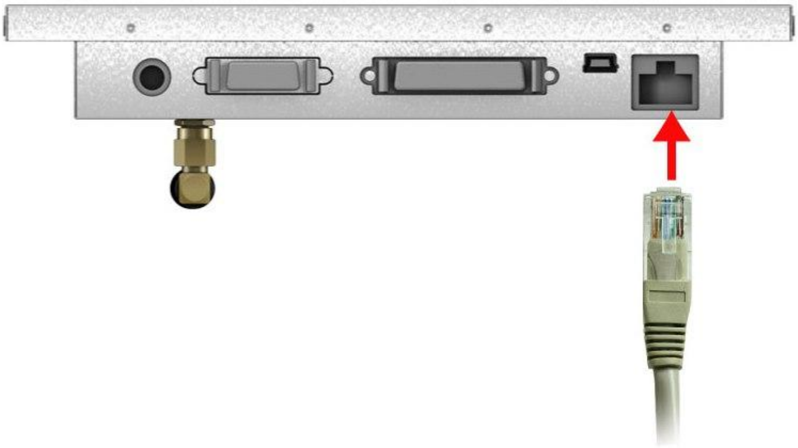
2.3 Connect Host USB

Users may connect host USB to any USB1.0 device, for example mouse, keyboard, USB storage, through a mini-USB cable as illustrated.

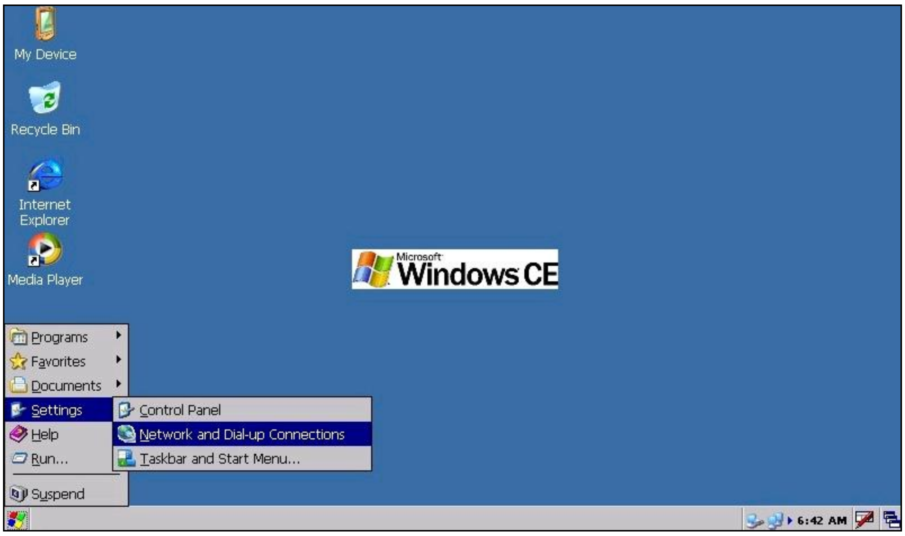


Connect to LAN

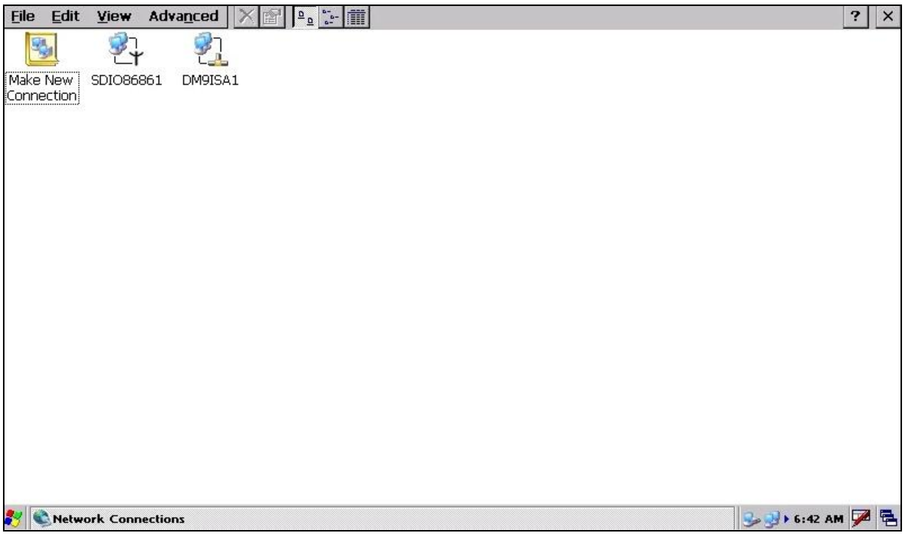
STEP1: Connect to LAN port as illustrated.



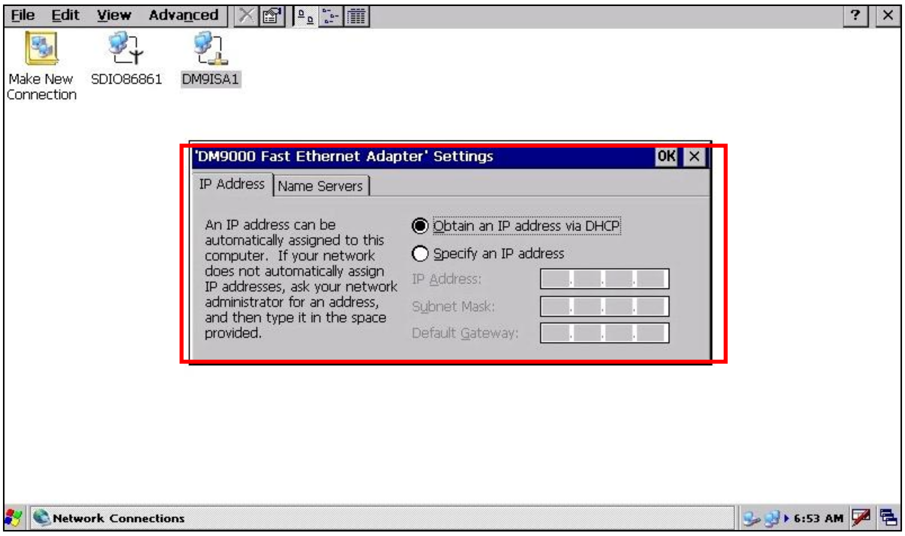
STEP2: Click on “Start-Setting-Networking and Dial-up connections” to set up a network



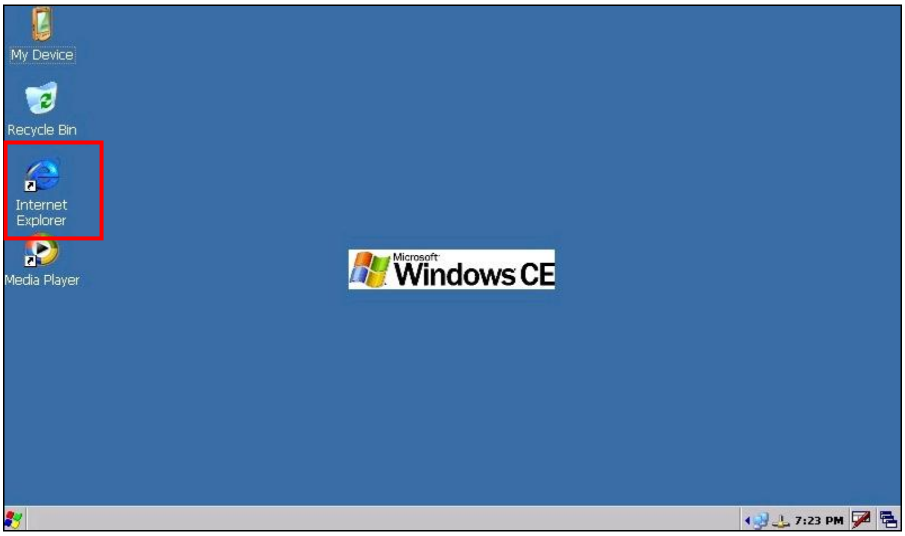
STEP3: Select ”DM9ISA1” to set up a LAN parameters.



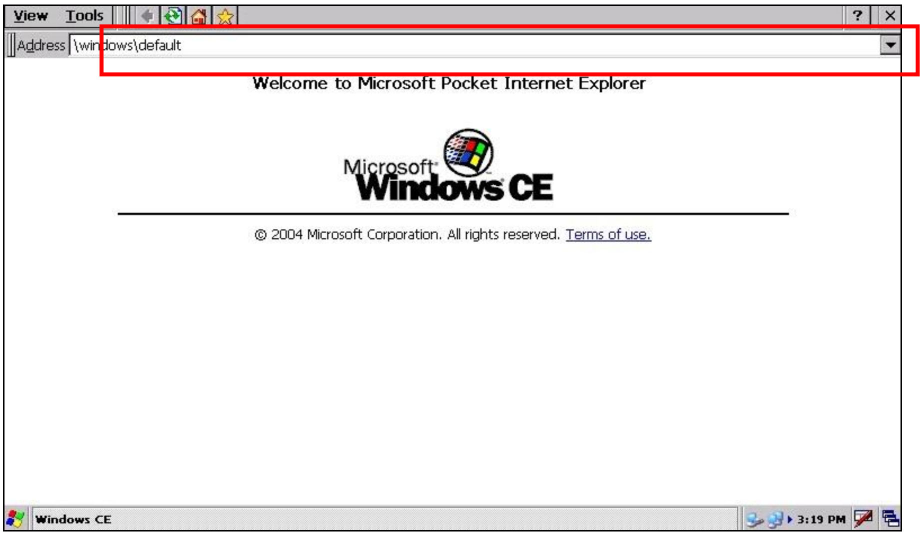
STEP4: Select static IP or DHCP IP as appropriate.



STEP5: Select Internet Explorer.



STEP6: Enter URL at the Address box

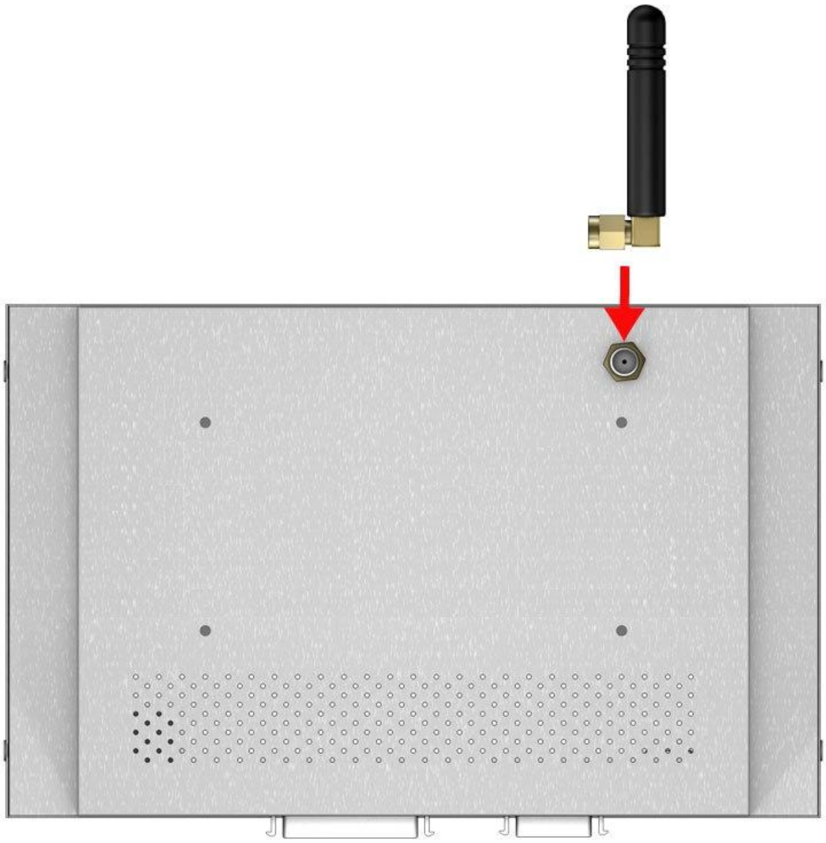


STEP7: Enter URL at the Address box

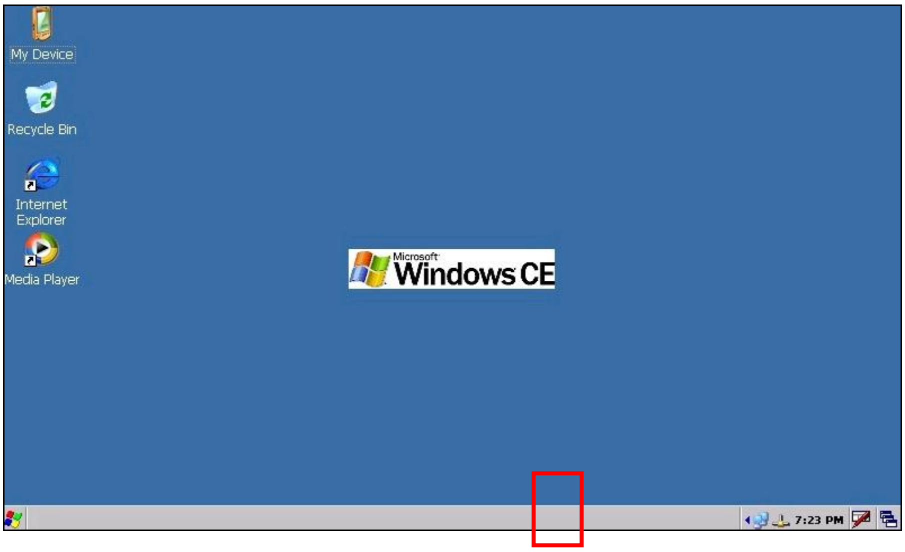


2.4 Connect Wi-Fi

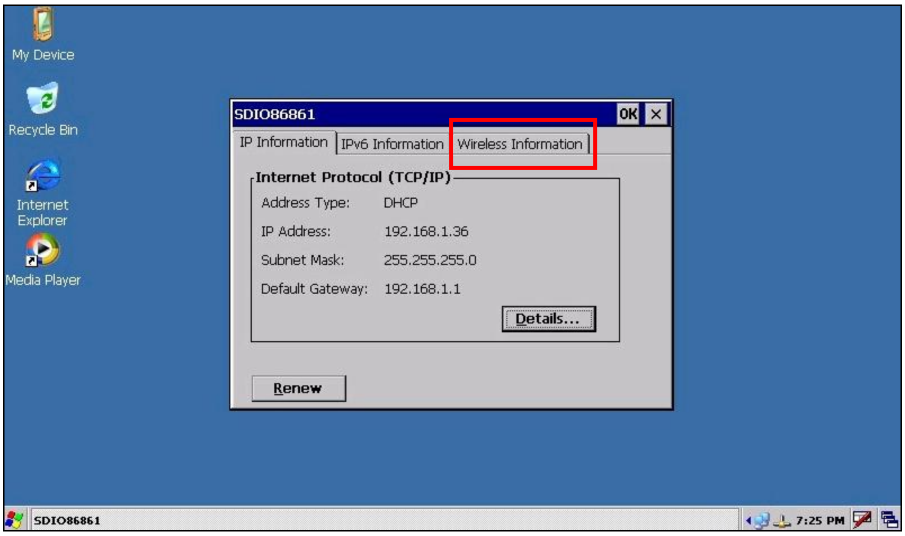
STEP1: Screw up the Antenna on BEGA220A as illustrated.



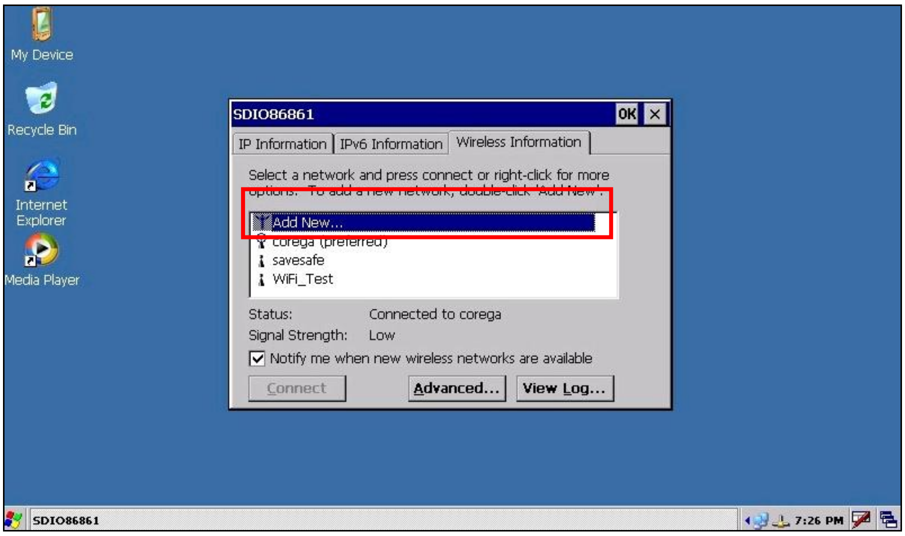
STEP2: Click on the network icon as highlighted to enter the wireless setup.



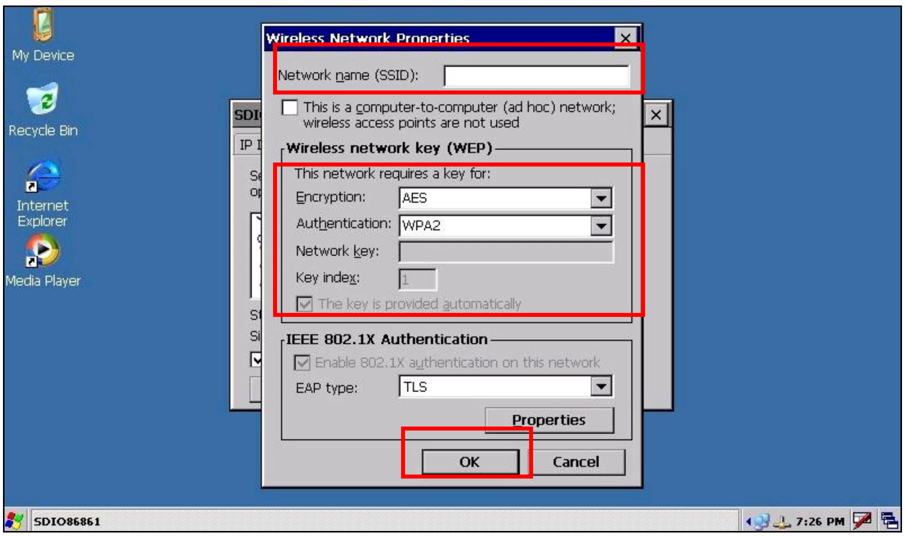
STEP3: Click on "Wireless Information" to set up wireless parameters



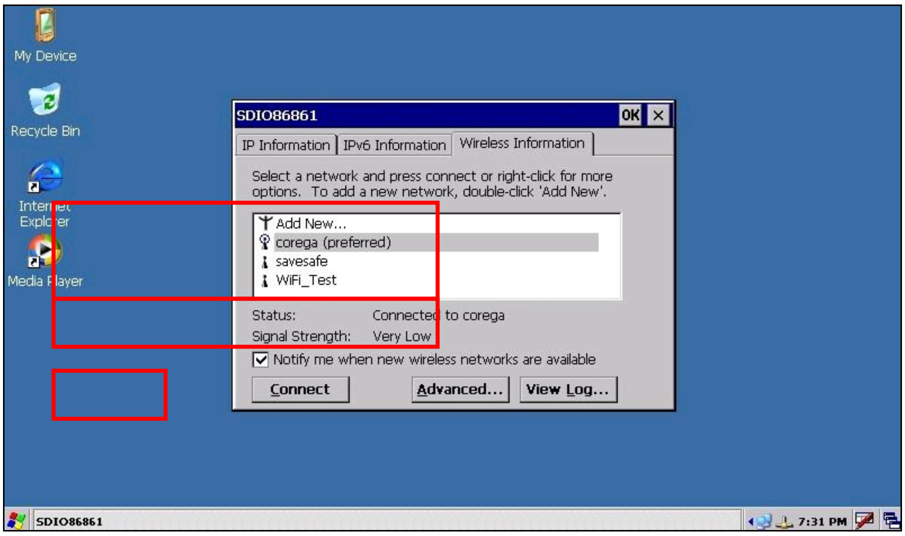
STEP4: Click on "Add New…" to add a new wireless connection



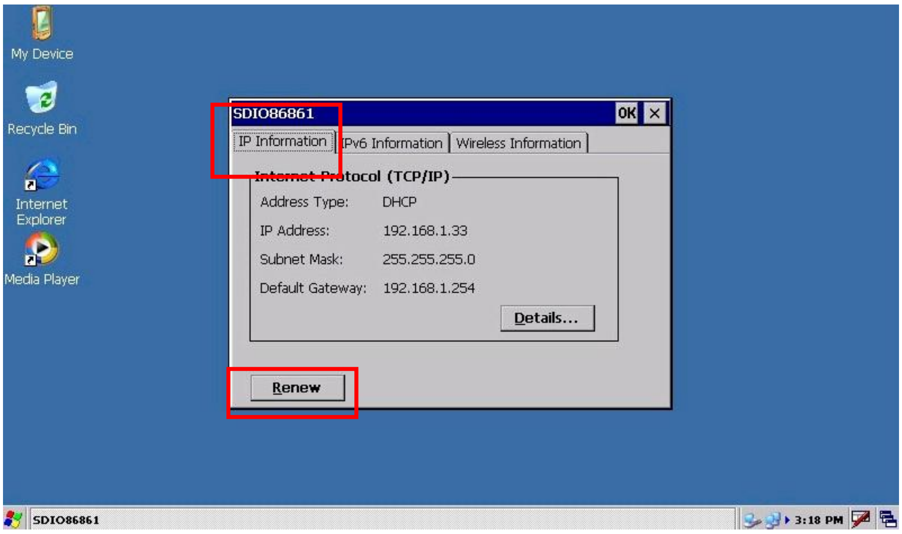
STEP5: Set up SSID, WEP, IEEE802.1X Authentications parameters as appropriate on BEGA220A, then click OK.



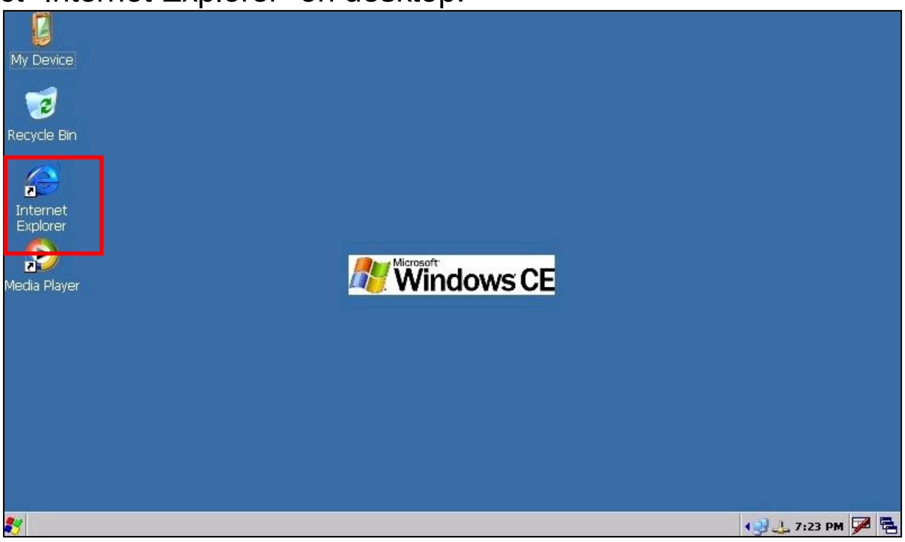
STEP6: Select the wireless access point, or SSID as appropriate, then click ”Connect” to enable wireless connection of BEGA220A. The connection status is shown right beneath the SSID selection box.



STEP7: Select ”IP Information” tab and click on “renew” to refresh IP address.



STEP8: Select “Internet Explorer” on desktop.



STEP9: Enter URL address in the "Address" box

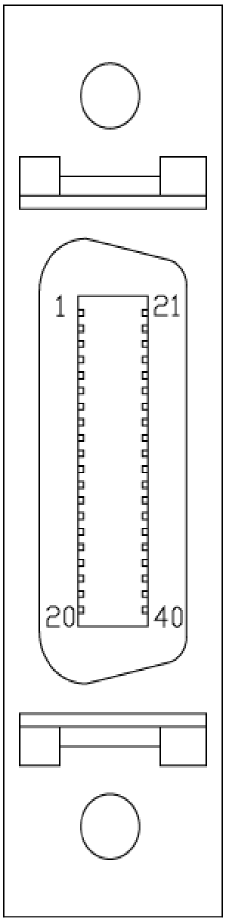


STEP10: then the browser will surf to the URL as specified.



2.5 Connect Communication Bus

2.5.1 Pin Assignment of Communication Bus



1	NC	Yellow/Red	21	GND	Green/White
2	NC	Yellow/Blue	22	GND	Blue/White
3	DSPC1	Black	23	DSPC2	White/Red
4	DSPC3	Brown	24	DSPC4	Brown/Black
5	NC	Brown/White	25	NC	Orange/Green
6	USBDP	Red	26	USBDN	Red/Black
7	NC	Brown/Green	27	GND	Orange/White
8	TXD1T	Orange	28	RXD1T	Orange/Black
9	RTS1T	Yellow	29	CTS1T	Yellow/Black
10	VBUS	Green	30	GND	Green/Black
11	TXD2T	Blue	31	RXD2T	Blue/Black
12	NC	X	32	GND	Red/White Red/Green
13	TXD3T	Purple	33	RXD3T	Purple/Black
14	VDD2	Grey	34	GND2	Orange/Blue
15	A-422R+	White	35	A-422R-	White/Black
16	A-422T+ A-485D+	Pink	36	A-422T- A-485D-	Pink/Black
17	VDD2	Grey/Black	37	GND2	Red/Blue
18	NC	X	38	GND	Purple/White Purple/Red
19	SS	Light green	39	SPIMOSI	Light green/Black
20	SPIMISO	Light blue	40	SPICLK	Light blue/Black

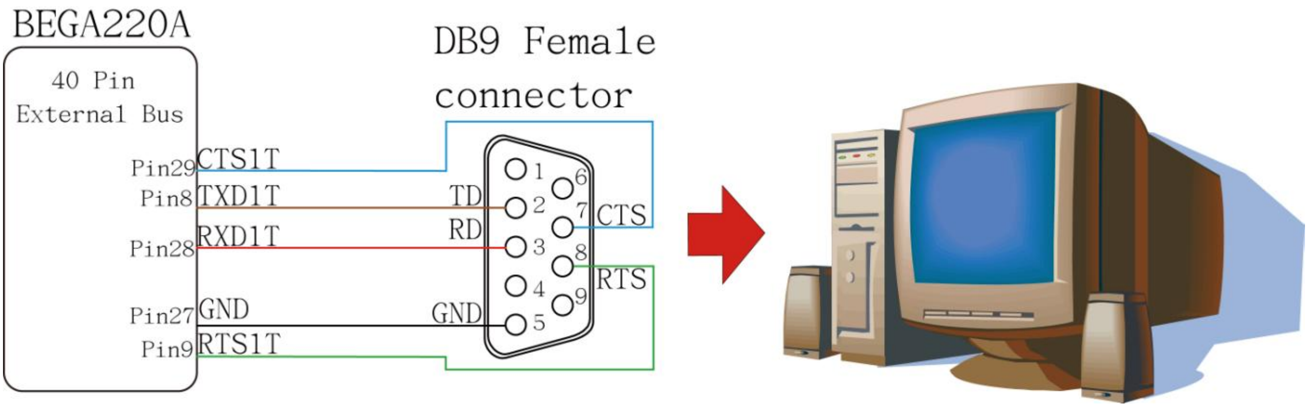
2.5.2 Serial Port Test

BEGA220A provides 3 sets of RS-232 for serial connection. There is also optional RS-485 or RS-422 interface available as options.

2.5.2.1 Connect Serial Port

1st set of RS-232 (Debug Port)(for update only)

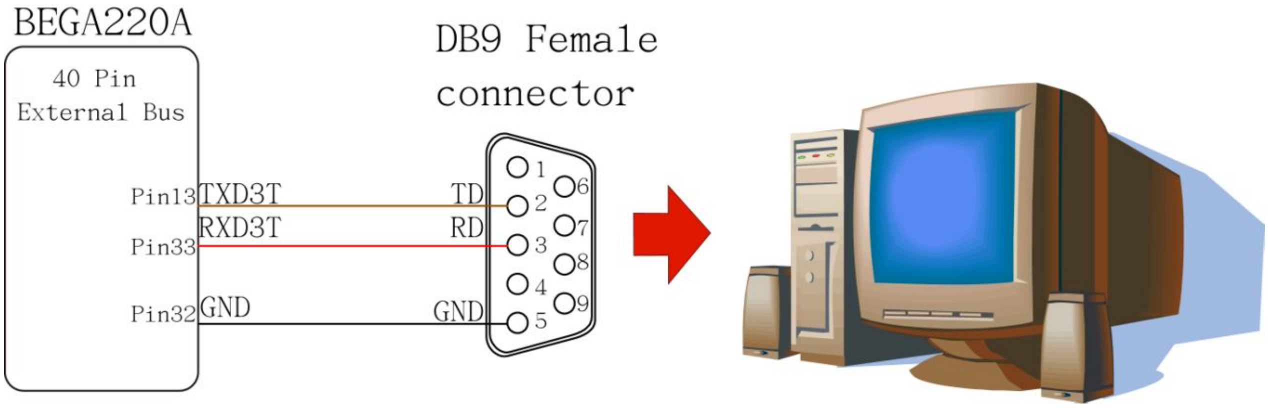
Connect 1st set of RS-232 on BEGA220A to DB9-female on PC's COM port. This RS-232 port is used as debug port ONLY with a signal level of +/-12v.



Pin Assignment of RS-232			
RS-232 DB9 Female		BEGA220A	
Pin No	Pin Name	Pin No	Pin Name
1	DCD	—	—
2	TD	8	TXD1T
3	RD	28	RXD1T
4	DTR	—	—
5	GND	27	GND
6	DSR	—	—
7	CTS	29	CTS1T
8	RTS	9	RTS1T
9	RI	—	—

2nd set of RS-232 (COM3)

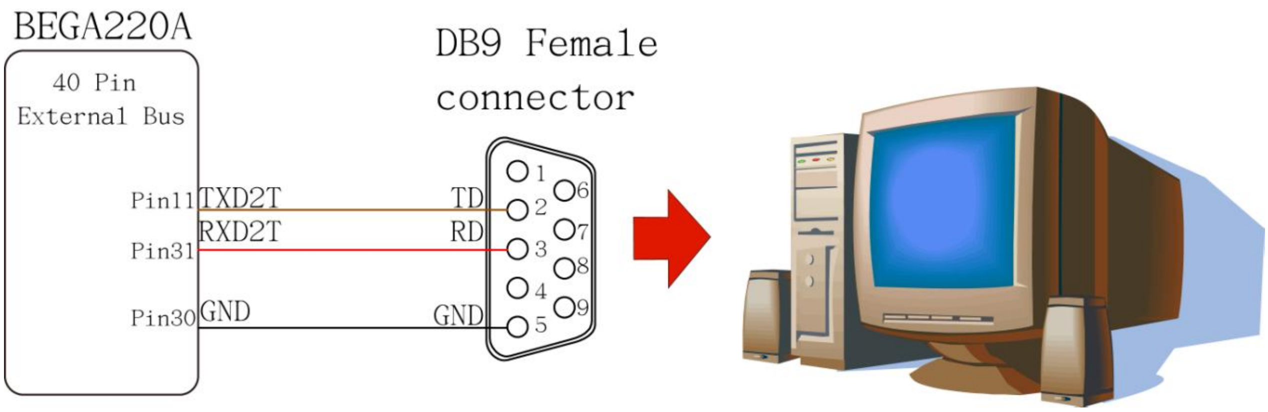
Connect 2nd set of RS-232 to PC's COM port. BEGA220A reserves COM3 port for the 2nd set of RS-232 and its signal level is at +/-12 v.



Pin Assignment of RS-232			
RS-232 DB9 Female		BEGA220A	
Pin No	Pin Name	Pin No	Pin Name
1	DCD	—	—
2	TD	13	TXD3T
3	RD	33	RXD3T
4	DTR	—	—
5	GND	32	GND
6	DSR	—	—
7	CTS	—	—
8	RTS	—	—
9	RI	—	—

3rd set of RS-232(COM4)

Connect 3rd set of RS-232, which is defaulted to COM4 on BEGA220A, to DB-9 COM port of PC. The signal level runs at +/-12v.



Pin assignment of RS-232			
RS-232 DB9 Female		BEGA220A	
Pin No	Pin Name	Pin No	Pin Name
1	DCD	—	—
2	TD	11	TXD2T
3	RD	31	RXD2T
4	DTR	—	—
5	GND	30	GND
6	DSR	—	—
7	CTS	—	—
8	RTS	—	—
9	RI	—	—

RS-422 (Option)(COM1)

BEGA220A supports 1 set of RS-422 serial interface and defaulted to COM1 port. It is supported to use a RS-232 to RS-422/485 converter board to connect to PC's COM port for connectivity test.

Pin Assignment of RS-422	
Pin No	Pin Name
35	A-422R-
15	A-422R+
16	A-422T+
36	A-422T-
34	GND2

*Note that either RS-422 or RS-485 can be used at a time.

*If no isolation is required, please connect Pin14 to Pin 12 and Pin 34 to Pin 32; otherwise, connect Pin 14 and 34 to a voltage of 5 +/- 10% voltage (4.75-5.25v).

RS-485 (Option)(COM1)

BEGA220A support one set of RS-485 serial interface and defaulted to COM1 port. During connectivity test, it is applicable to use a RS-232 to RS-422/485 converter board to connect to PC's COM port.

Pin assignment of RS-485	
Pin No	Pin Name
37	GND2
16	A-485D+
36	A-485D-

*Note that either RS-422 or RS-485 can be used at a time.

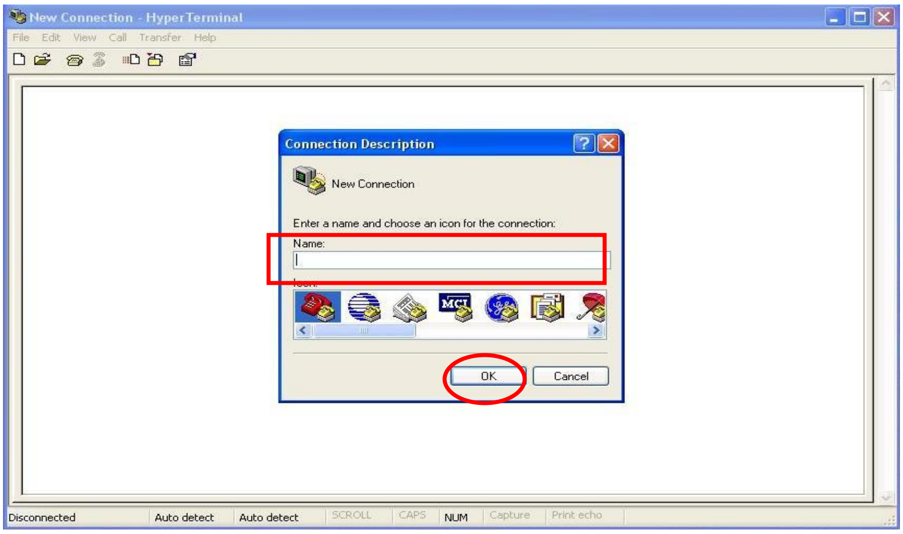
*If no isolation is required, please connect Pin14 to Pin 12 and Pin 34 to Pin 32; otherwise, connect Pin 14 to a voltage of 5 +/- 10% voltage (4.75-5.25v), and pin 34 to ground.

2.5.2.2 Serial Port test procedure

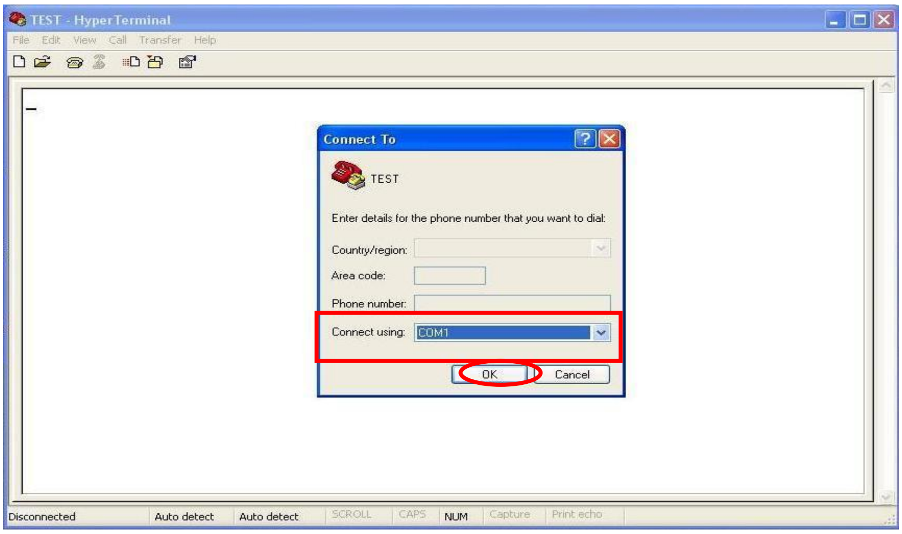
To test serial connectivity , first connect the RS-232 cable as appropriate, and run the hyper-terminal program to test if receiving and sending function normally. Here is the step guide:

PC set up:

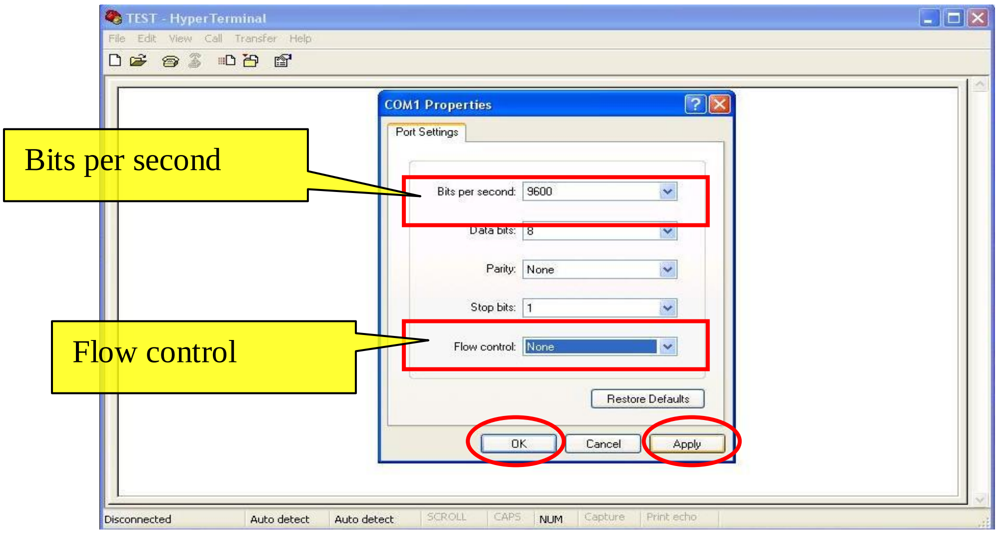
STEP1: Run "hyper-terminal" program under start-program-accessories-communication", then enter a name for this session and click ok.



STEP2: 在 Select PC's COM port in "Connect using" box to connect to serial port on BEGA220A. Then click OK.

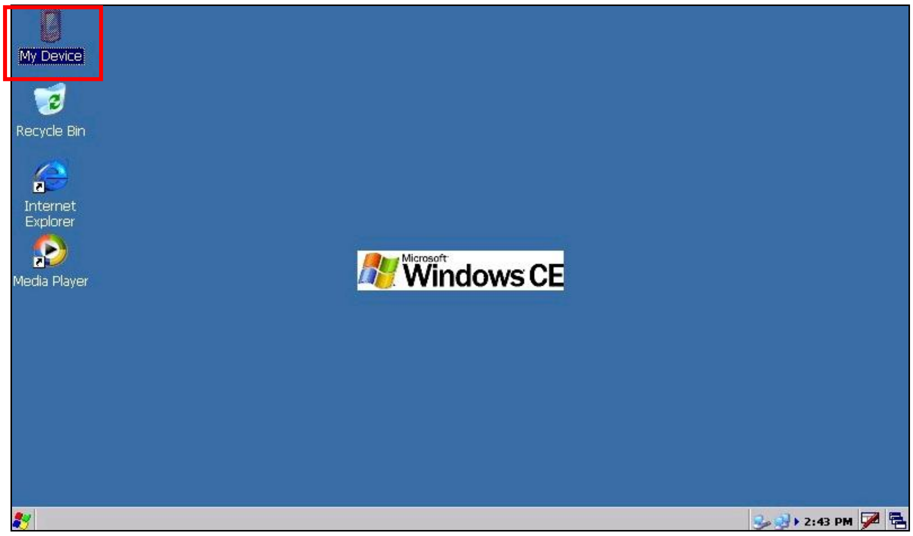


STEP3: Pull down "Bits per second" to set up communication speed (usu. Ranges from 9600 to 115200). Make sure the baud rate value is identical to that on BEGA220A. Select None for "Flow control", then click ""Apply", and click "OK".

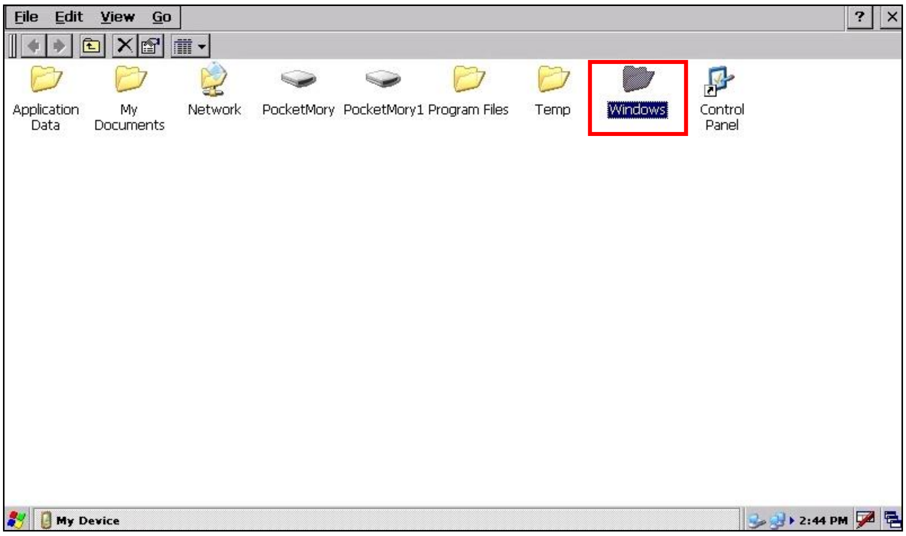


BEGA220A set up procedure:

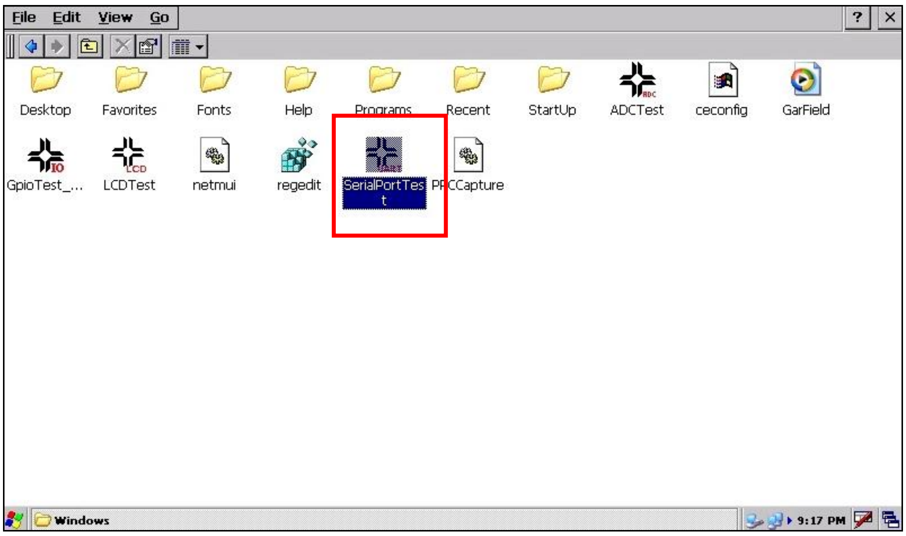
STEP1: Select ”My Device” on BEGA220A



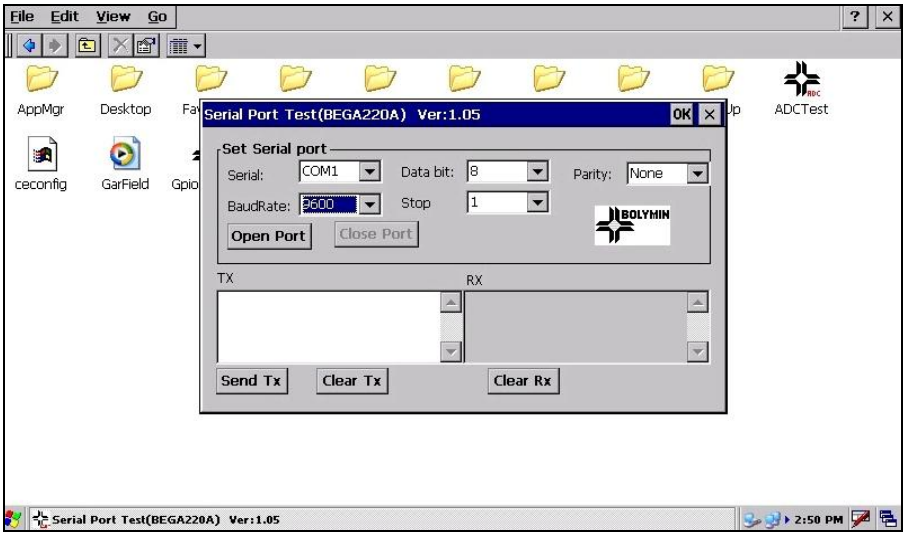
STEP2: Select “Windows” folder



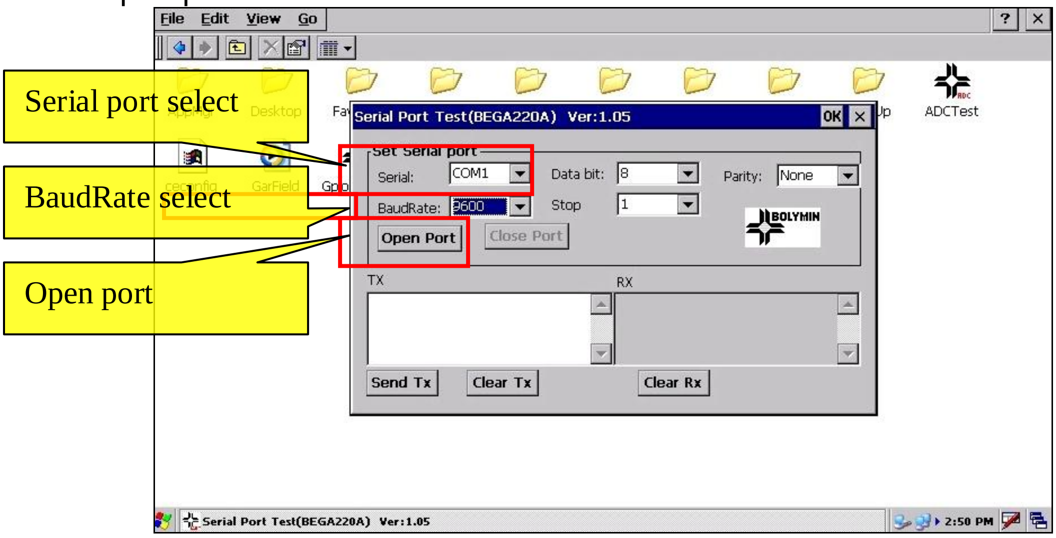
STEP3: Run”SerialPortTest”



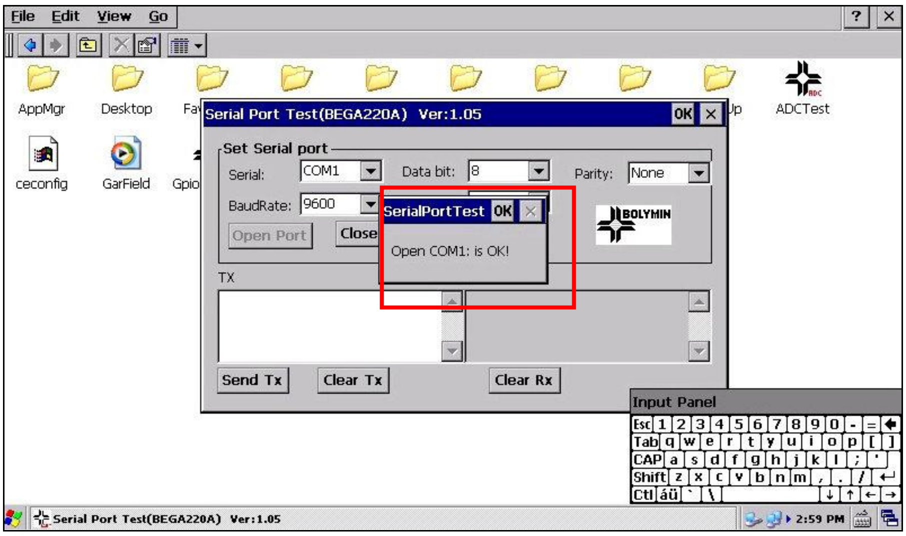
STEP4: " Here is the initial screen of SerialPortTest"



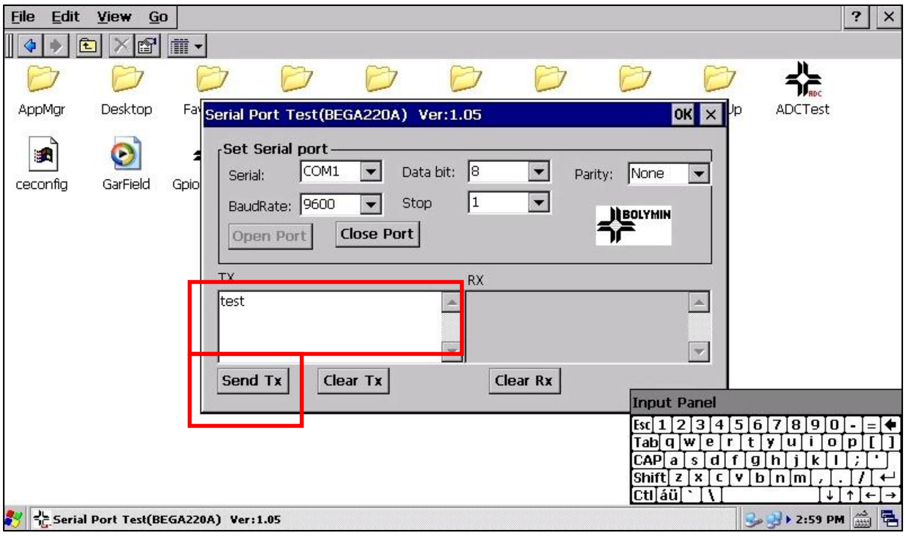
STEP5: Select COM port as appropriate. Pull down at Baud Rate ranged between 9600 to 115200 and use the same value at both BEGA220A and PC end. Then click on "Open port" to activate a serial connection.



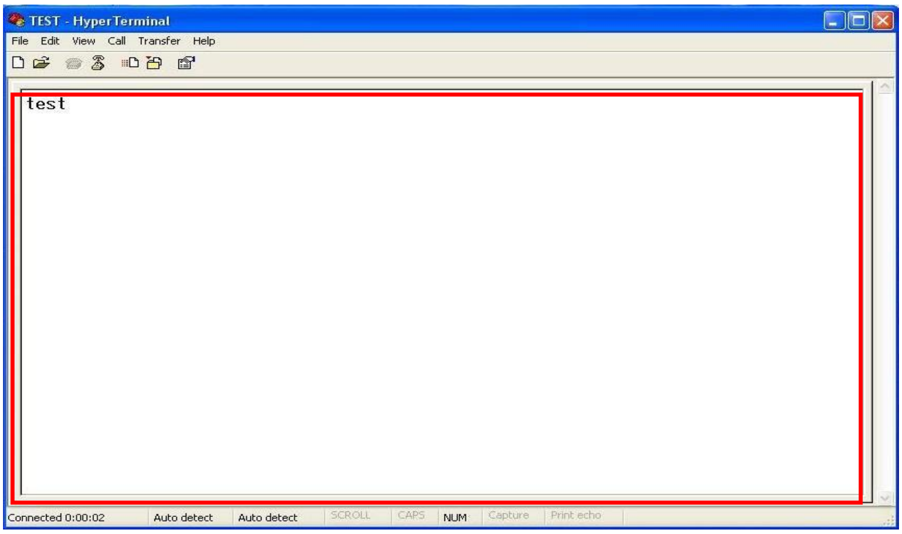
STEP6: If everything goes right, you will see the pop-up window as follows. Click on OK to proceed.



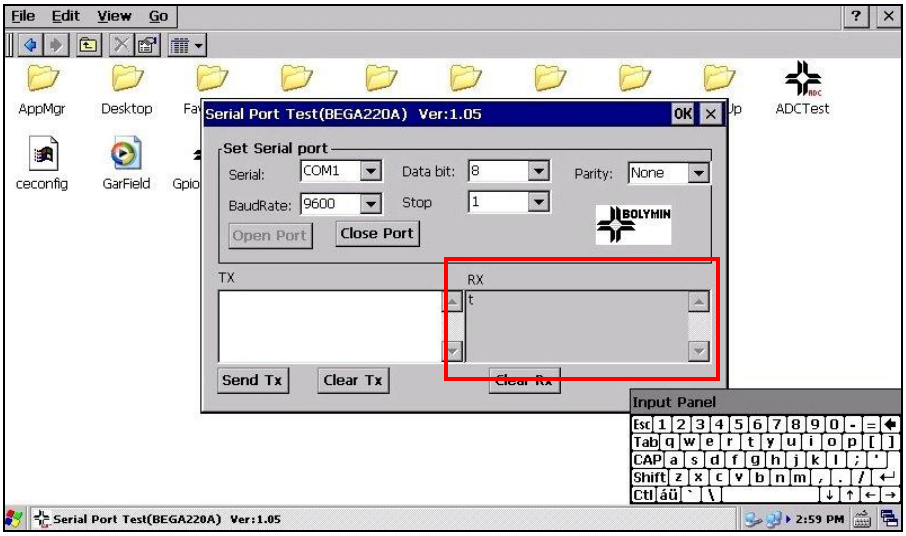
STEP7: Enter “text” at Tx” window and click on”Send Tx” button.



STEP8: At PC end, you will see the same text entered echoed back.

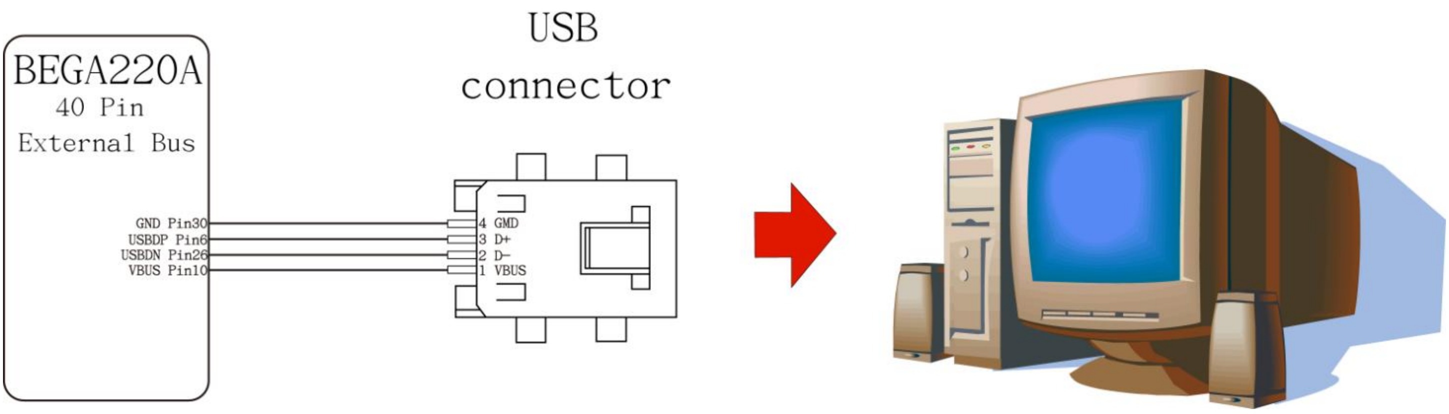


STEP9: Enter some text at PC end and note an echo of those text will appear at RX window at BEGA220A end.



2.5.3 Device USB installation

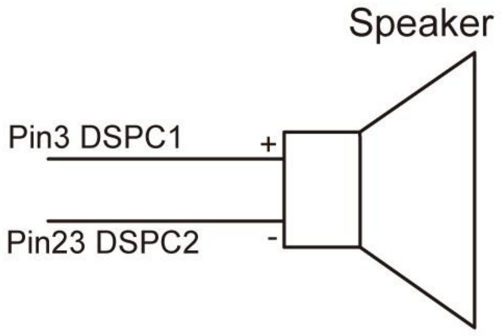
BEGA220A supports one set of Device USB to communicate with a PC through ActiveSync software for data sync. Here illustrates the connection of Device USB. Please refer to chapter 3.1.1 for ActiveSync operation.

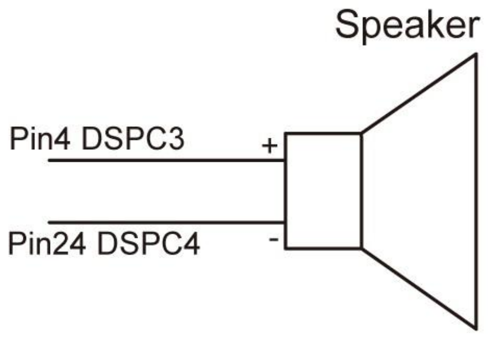


Pin assignment of Device USB			
USB		BEGA220A	
Pin No	Pin Name	Pin No	Pin Name
1	VBUS	10	VBUS
2	D-	26	USBDN
3	D+	6	USBDP
4	GND	30	GND

2.5.4 Speaker installation

BEGA220A offers 2 sets of speaker circuits to connect to 8Ω/2W speaker. Connect speakers as illustrated to support audio playback.





1st set of Speaker

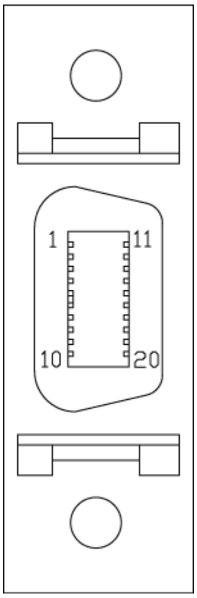
Pin Assignment of Speaker			
Speaker		BEGA220A	
Pin No	Pin Name	Pin No	Pin Name
1	+	3	DSPC1
2	-	23	DSPC2

2nd set of peaker

Pin Assignment of Speaker			
Speaker		BEGA220A	
Pin No	Pin Name	Pin No	Pin Name
1	+	4	DSPC3
2	-	24	DSPC4

2.6 ADC_GPIO Bus

2.6.1 Pin Assignments of ADC_GPIO Bus



1	ADCT0	Brown	11	ADCT3	Blue
2	ADCT1	Orange	12	ADCT4	Blue/Black
3	ADCT2	Brown/Black	13	ADCT5	Purple
4	GND	Orange/Black	14	GND	White
5	KEY1	Red	15	KEY6	Purple/Black
6	KEY2	Yellow	16	KEY7	White/Black
7	KEY3	Red/Black	17	KEY8	Grey
8	KEY4	Yellow/Black	18	KEY9	Light Blue
9	KEY5	Green	19	KEY10	Grey/Black
10	KEY11	Green/Block	20	KEY12	Light blue/Black

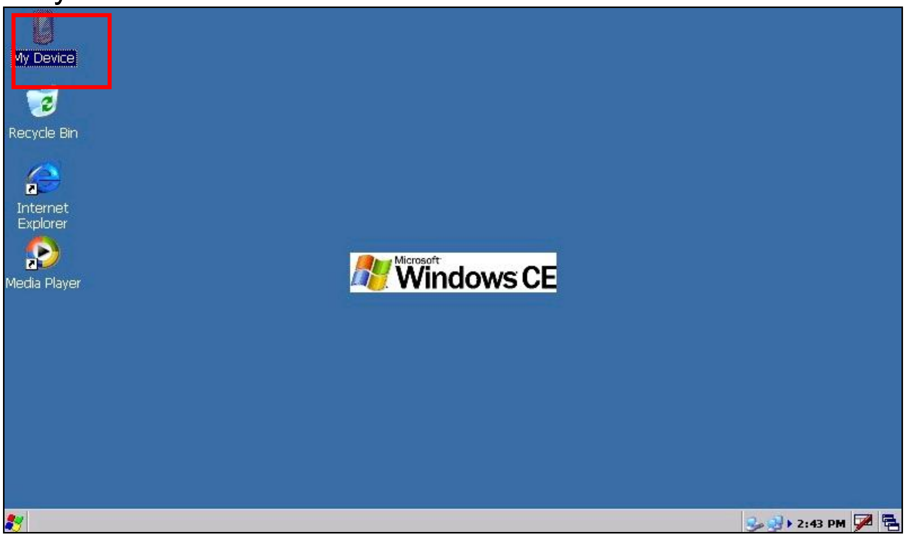
2.6.2 GPIO Test

BEGA220A offers 12 general purpose programmable I/O ports (GPIO). Please refer to sample code at chapter 4.4.The typical signal length is around 10 to 15 cm and the rated voltage is as follows

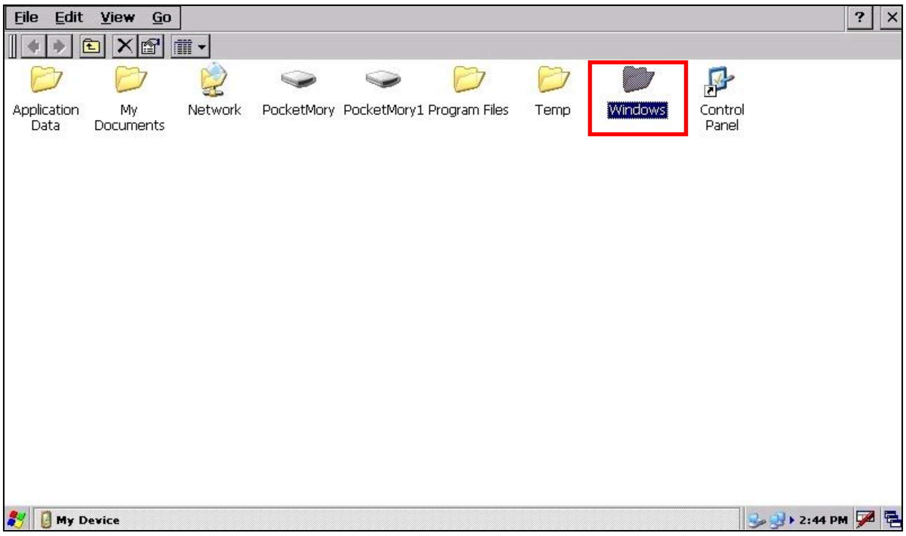
Item	Symbol	Min	Typ	Max	Unit
High Level Input Voltage	VIH	2.3	—	3.6	V
Low Level Input Voltage	VIL	-0.3	—	09	V
High Level Output Voltage	VOH	3.1	—	3.3	V
Low Level Output Voltage	VOL	—	—	0.2	V

Test Procedure:

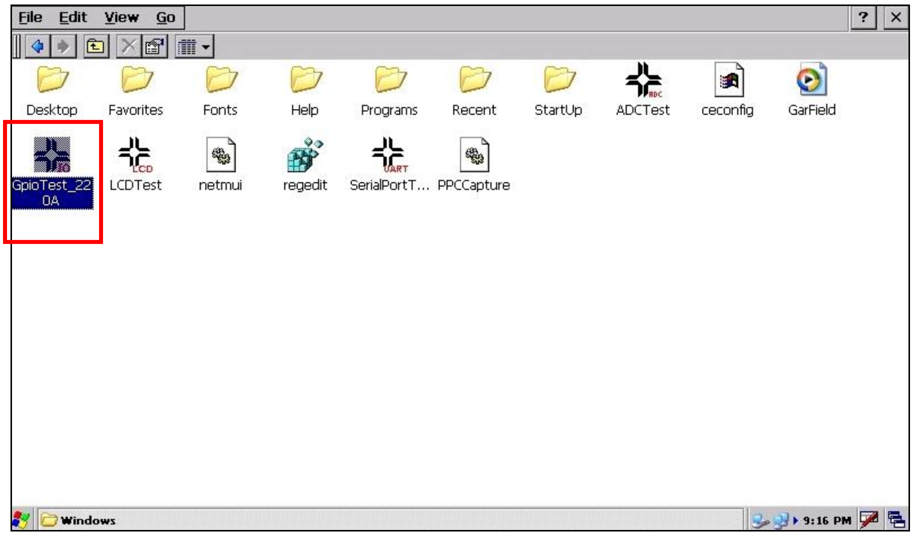
STEP1: Select"My Device" on BEGA220A



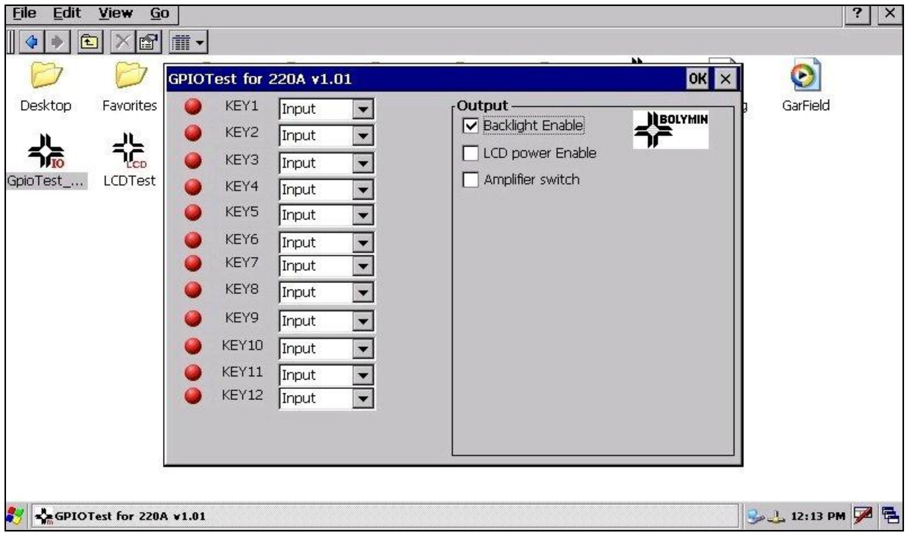
STEP2: Select "Windows" folder.



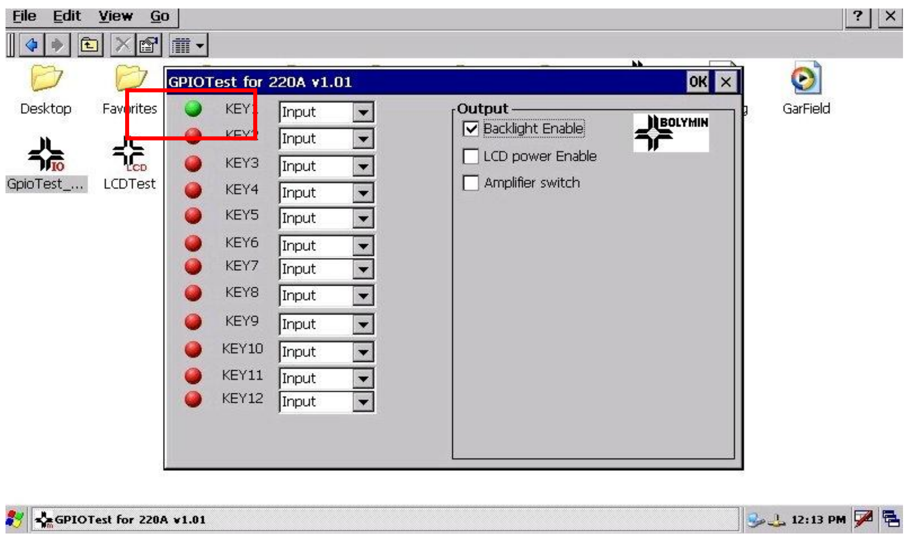
STEP3: Run ”GpioTest_220A”



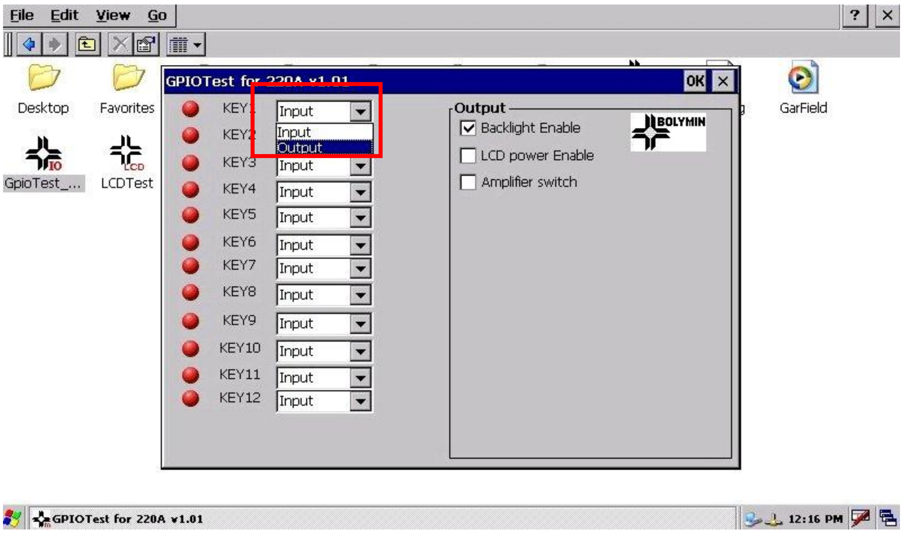
STEP4: Here is the initial screen of ”GpioTest_220A” and defaulted to test all KEYs as inputs.



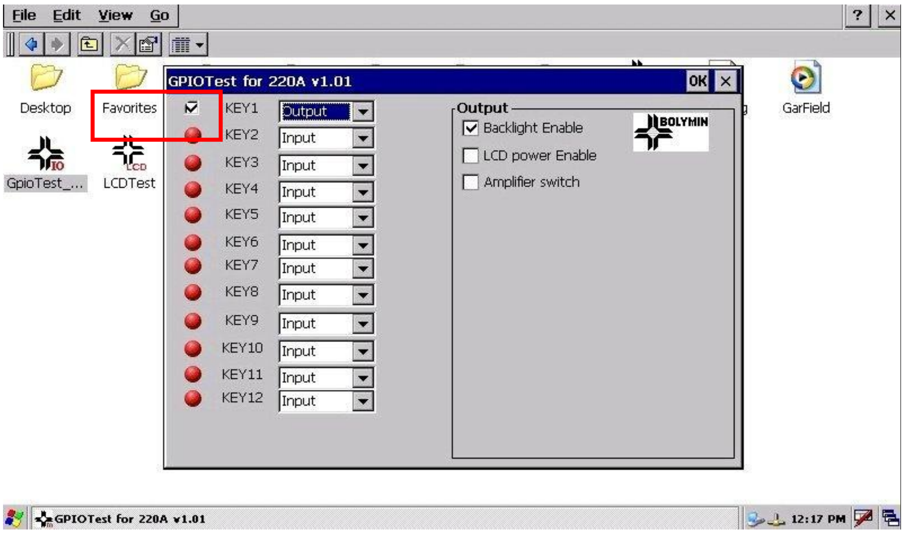
STEP5: A green light at KEY stands for a high input signal



STEP6: To test output, pull down the Input and change the KEY to output.



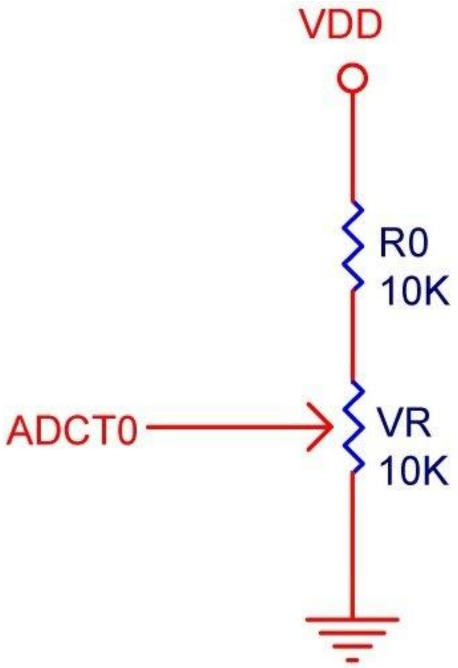
STEP7: While KEY is defined as OUTPUT as step6, check on red circle to the left of KEYto generate a high output voltage.



2.6.3 ADC test

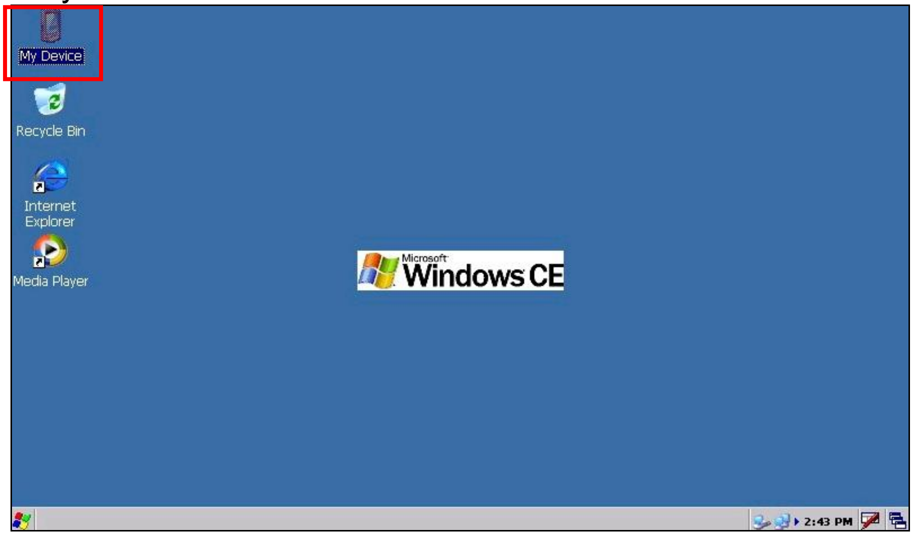
2.6.3.1 ADC test set-up

Prior to ADC port testing, connect ADCT0 port as illustrated. Use a Vdd of 3.3 volt and voltage divider to limit maximum ADCT0 to be 1.6 volt (ADC readout as 4095 , 12-bit unsigned). Then run the ADC test by using test program as BEGA220A provided. Note that the maximum voltage of ADCTO should not exceed 1.6 volt.

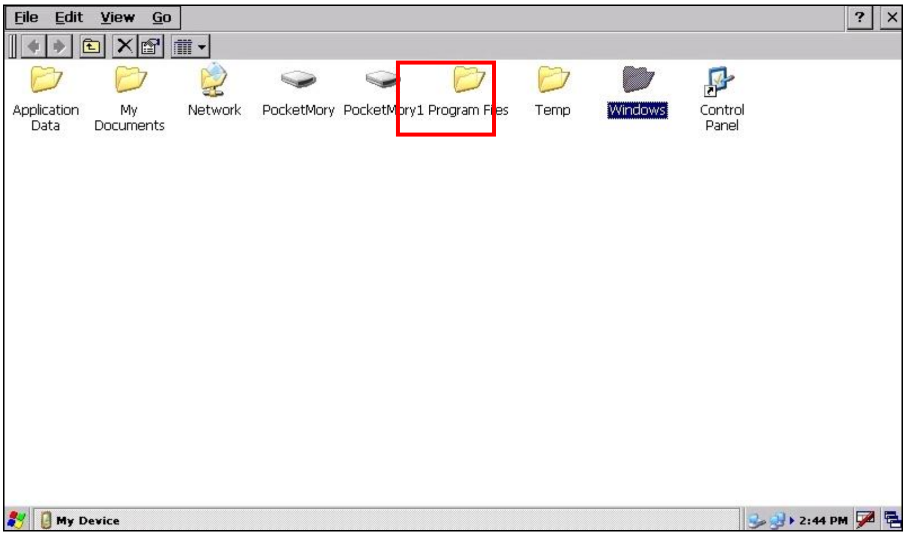


2.6.3.2 ADC test procedure

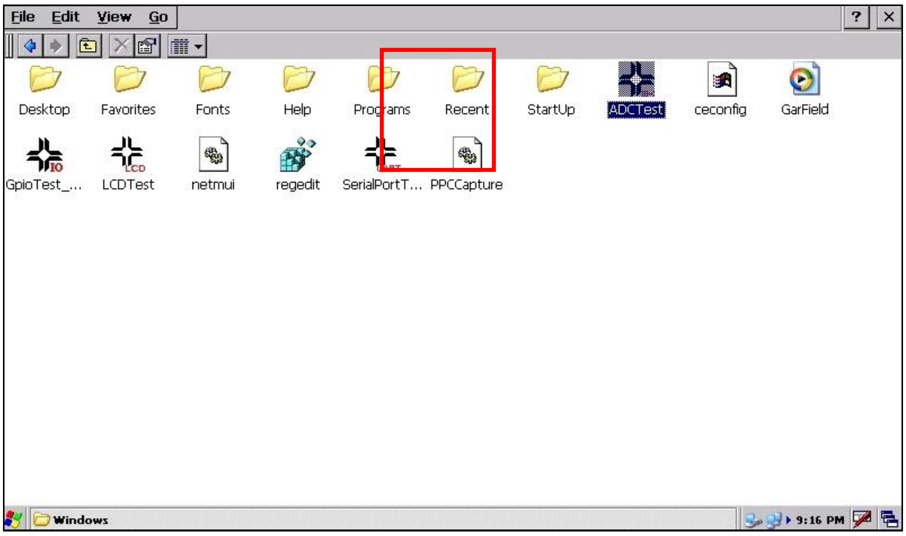
STEP1: Select"My Device"on BEGA220A



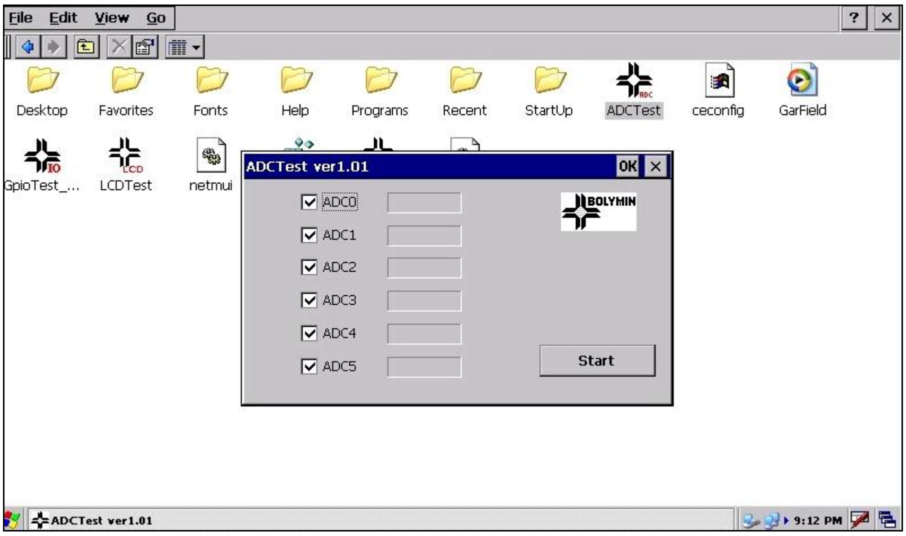
STEP2: select "Windows" folder



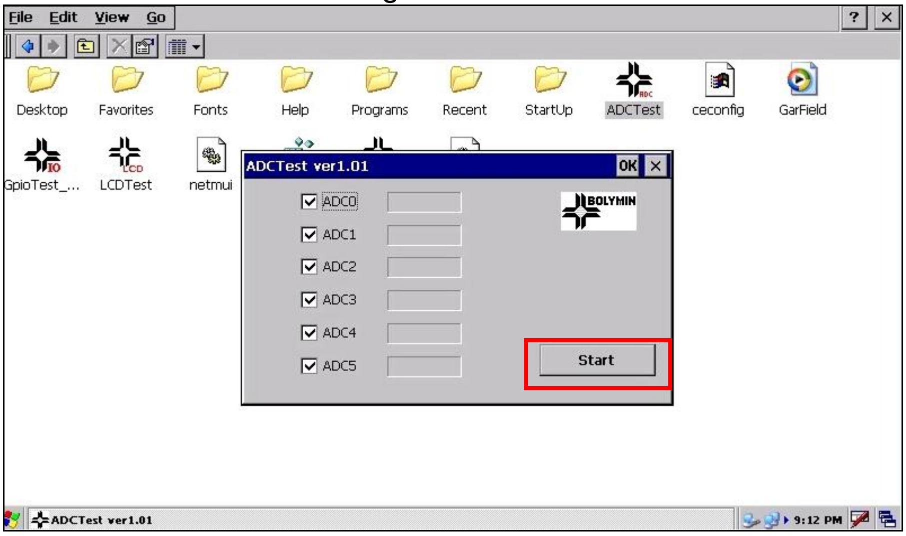
STEP3: Run "ADCTest" program by double-clicking the icon.



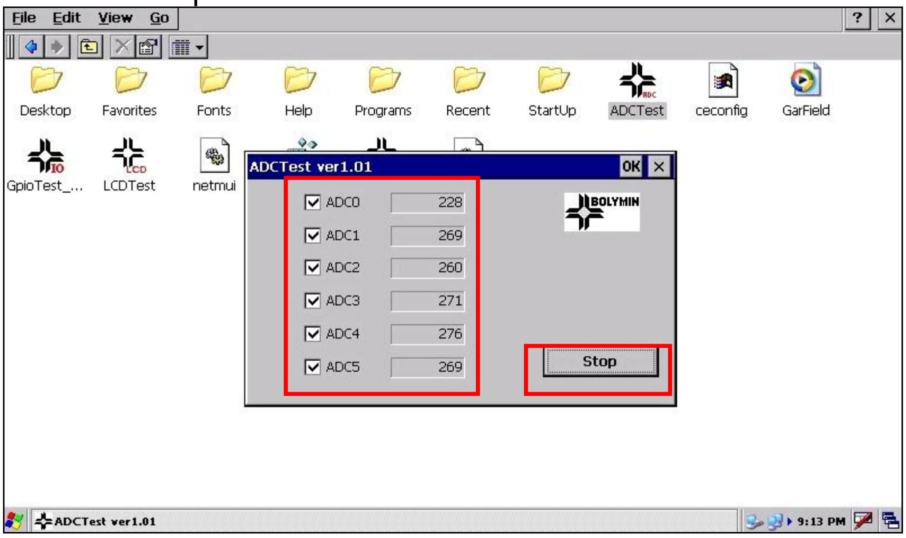
STEP4: "ADCTest" initial screen is as follows:



STEP5: Click "Start" button to start testing ADC.



STEP6: Adjust variable resistor (VR) and observe the changes in read-out of ADC Port value. Click on "Stop" to finish the ADC test.



3 BEGA220A Programming Guide

This chapter demonstrates connection from BEGA20A to PC and how to use software to control serial port, GPIO , ADC, Backlight, and SPI. Chapter 3 consists of the following:

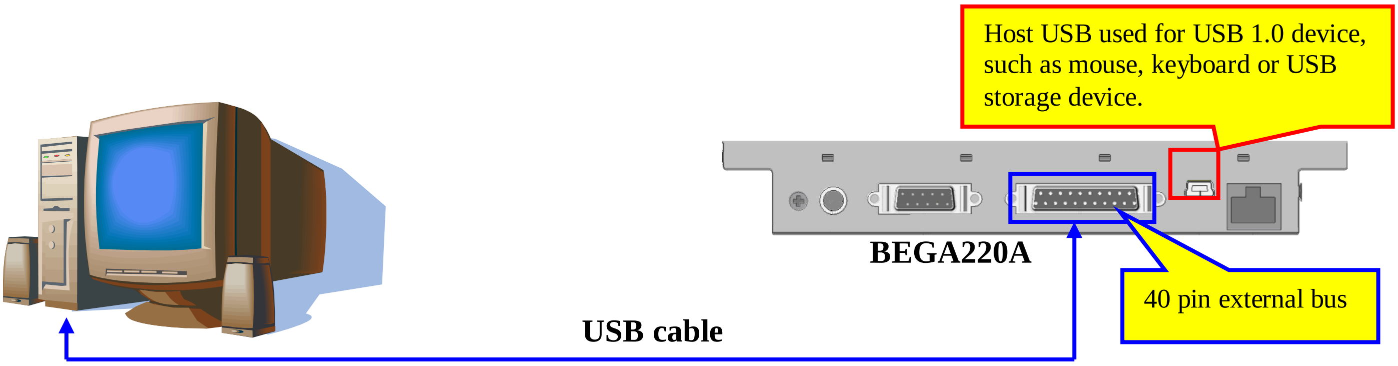
3.1 Transfer File Between BEGA220A and PC

3.1.1 Connect PC and BEGA220A

User may setup the connection between desktop PC and BEGA220A by following steps:

STEP 1. Install Microsoft ActiveSync 4.5 on desktop PC. You may download ActiveSync from:
http://www.microsoft.com/downloads/details.aspx?familyid=9e641c34-6f7f-404d-a04b-dc09f8141141&displaylang=en&tm
After installation, you need to restart PC.

STEP 2. Connect desktop PC and BEGA220A by USB cable. Below picture shows the hardware connection between desktop PC and BEGA220A for file transfer.

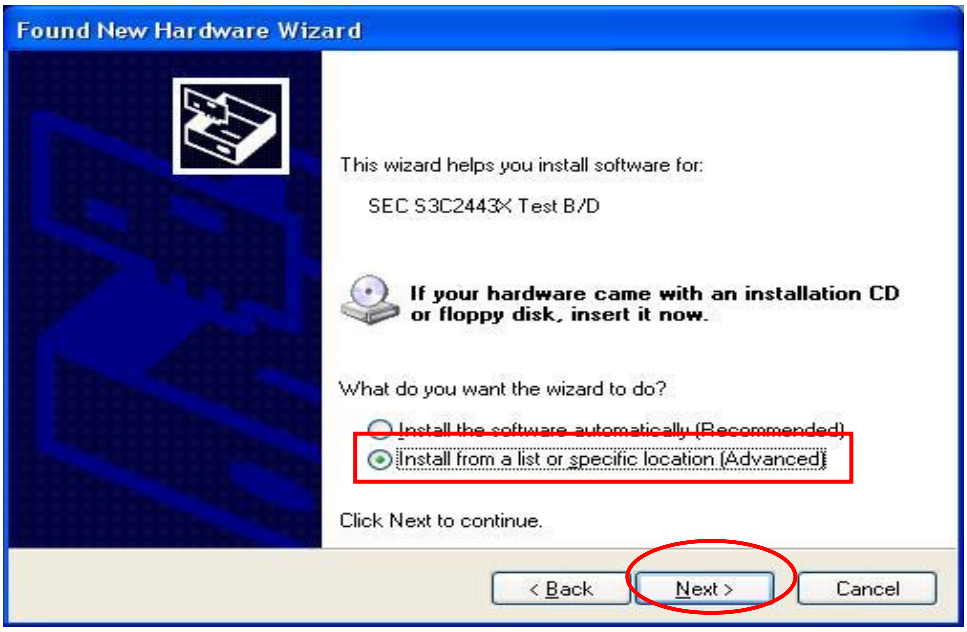


Please note that **the USB cable should plug into the connector of device USB wiring from the 40 pin communication bus**. Below table shows the pin assignment of device USB on the 40 pin external bus.

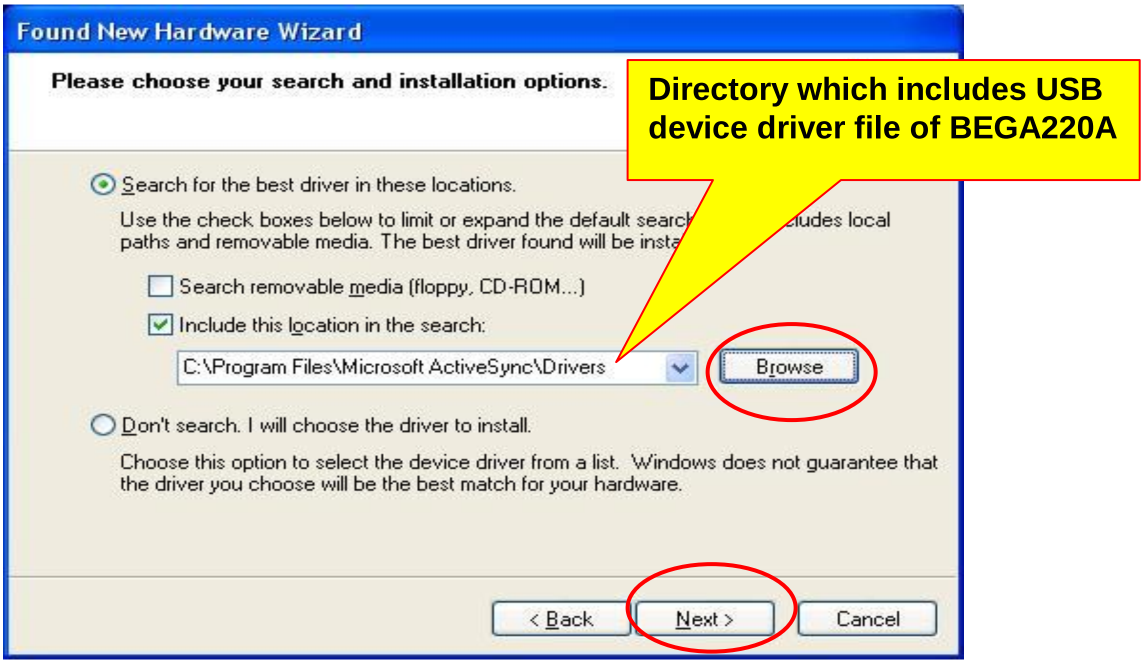
USB connector		40 pin communication bus of BEGA220A	
Pin No	Pin Name	Pin No	Pin Name
1	VBUS	10	VBUS
2	D-	26	USBDN
3	D+	6	USBDP
4	GND	30	GND

STEP 3. Power on BEGA220A and connect BEGA220A and PC by USB cable. For the first connection, windows system on PC will request the USB device driver of BEGA220A. Please install USB driver by following procedures.

(1). Select the advance item on below dialog and click "Next" button.



(2). Click "Browse" button and then select the directory which includes USB device driver file of BEGA220A. Click "Next" button.



*The following download URL contain all needed driver for 220A Active Sync/Samgsung drivers: http://www.bolymin.com.tw/manual/dnw.rar

(3). Click “Continue” button



(4). Click “Finish” button. Now BEGA220A may connect to PC by ActiveSync.

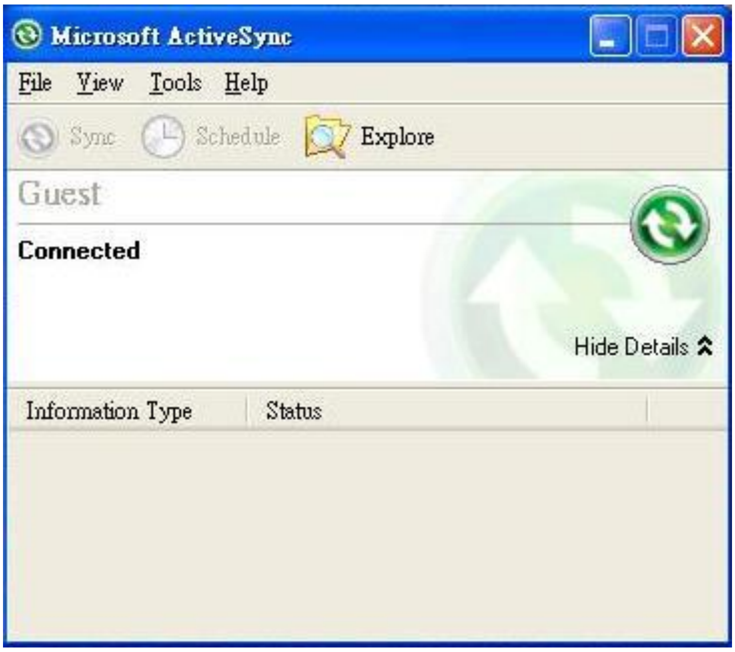


(5). Select “No” and click “Next” button to cancel the synchronization.

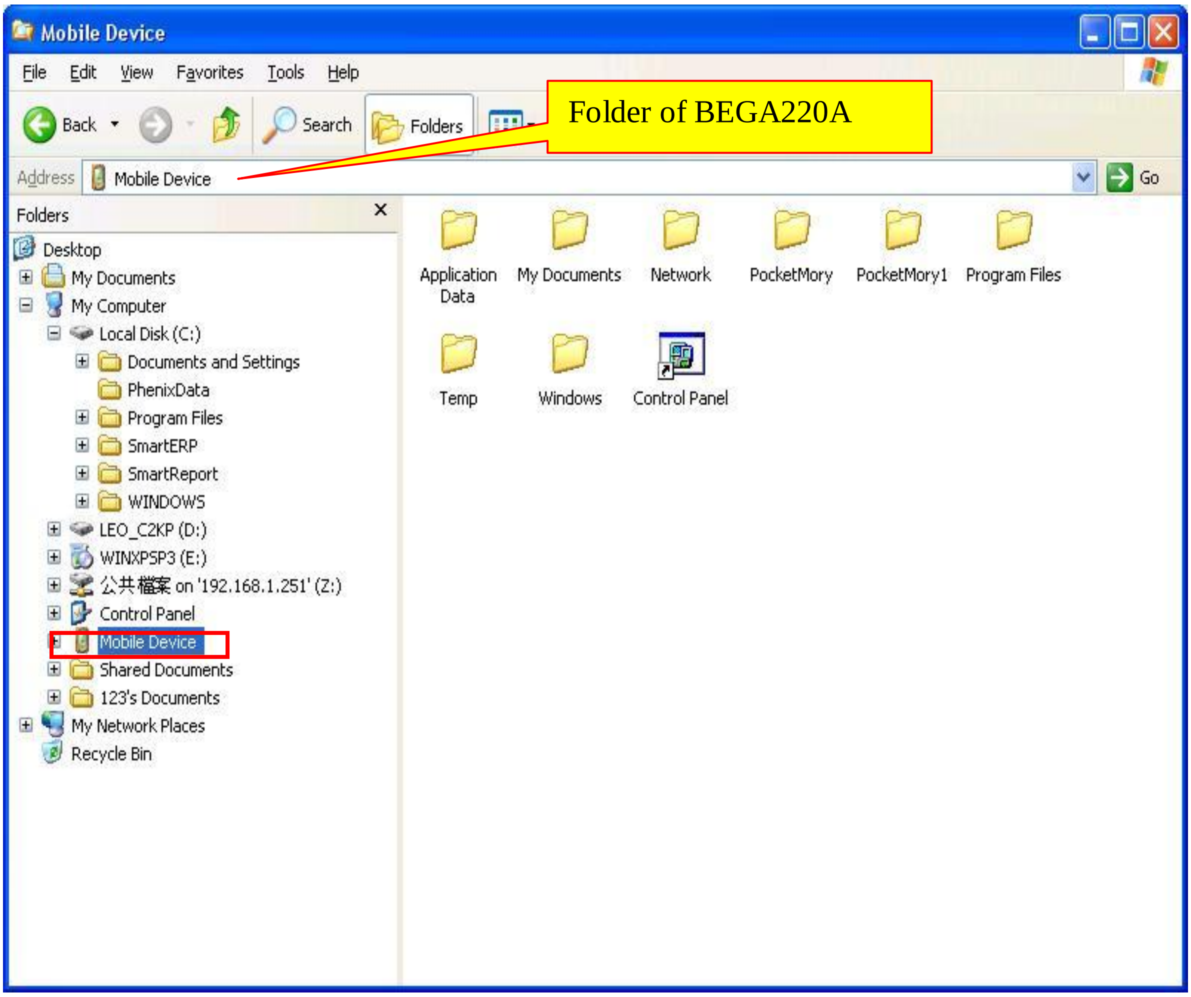


3.1.2 Transfer Files

After connecting PC and BEGA220A, below screen will display. The green cycle means the connection between PC and BEGA220A has been built.



Execute "Explore" program and move into the folder of BEGA220A, you can transfer files between PC and BEGA220A.



3.2 Programming for BEGA220A

3.2.1 Setup Development environment

By following steps, we can setup the development environment for WinCE 5.0:

1. **Install Microsoft eMbedded Visuall C++ 4.0(eVC 4.0) into desktop PC :** eVC 4.0 can be downloaded from http://www.microsoft.com/downloads/details.aspx?FamilyID=1DACDB3D-50D1-41B2-A107-FA75AE960856&displayLang=en.
   Use free serial number : TRT7H-KD36T-FRH8D-6QH8P-VFJHQ

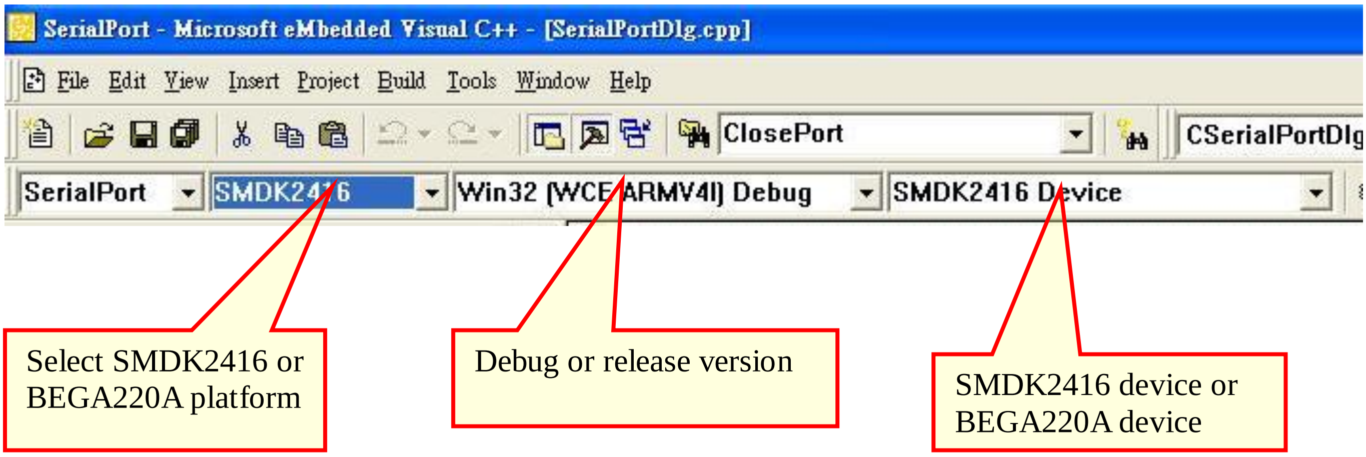
2. Install service pack 4 for eVC 4.0 and here is the download URL: http://download.microsoft.com/download/a/7/3/a735c7fb-dcbd-429f-9090-d09b3b15d3fa/evc4sp4.exe
   After the patch, the eVC version is 4.00.1610.0.

3. **Connect BEGA220A and Desktop PC by procedures in section 3.1.2**

4. **Install SDK of BEGA220A provided by Bolymin.** The installation file may be found in the product CD. Here is the download URL: http://www.bolymin.com.tw/manual/BEGA220A_INX_SDK_100804_093.msi

5. **The platform setting of embedded Visual C++:**

Following pictures show the required setting of eVC 4.0::

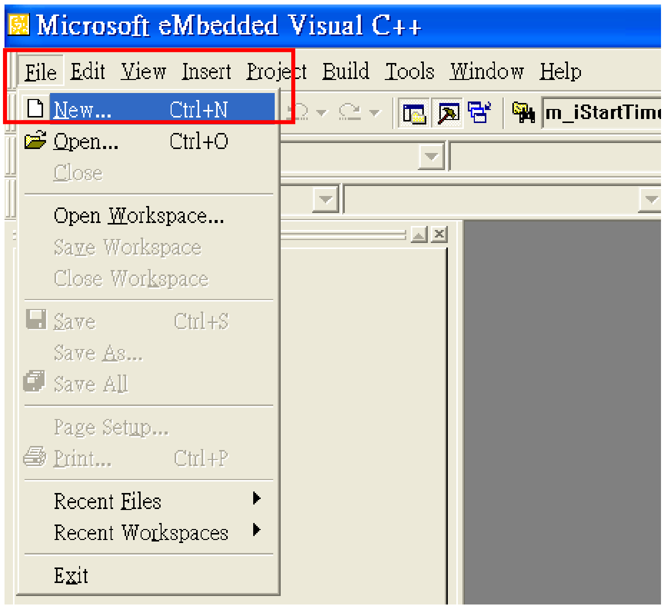


3.2.2 Create New Project

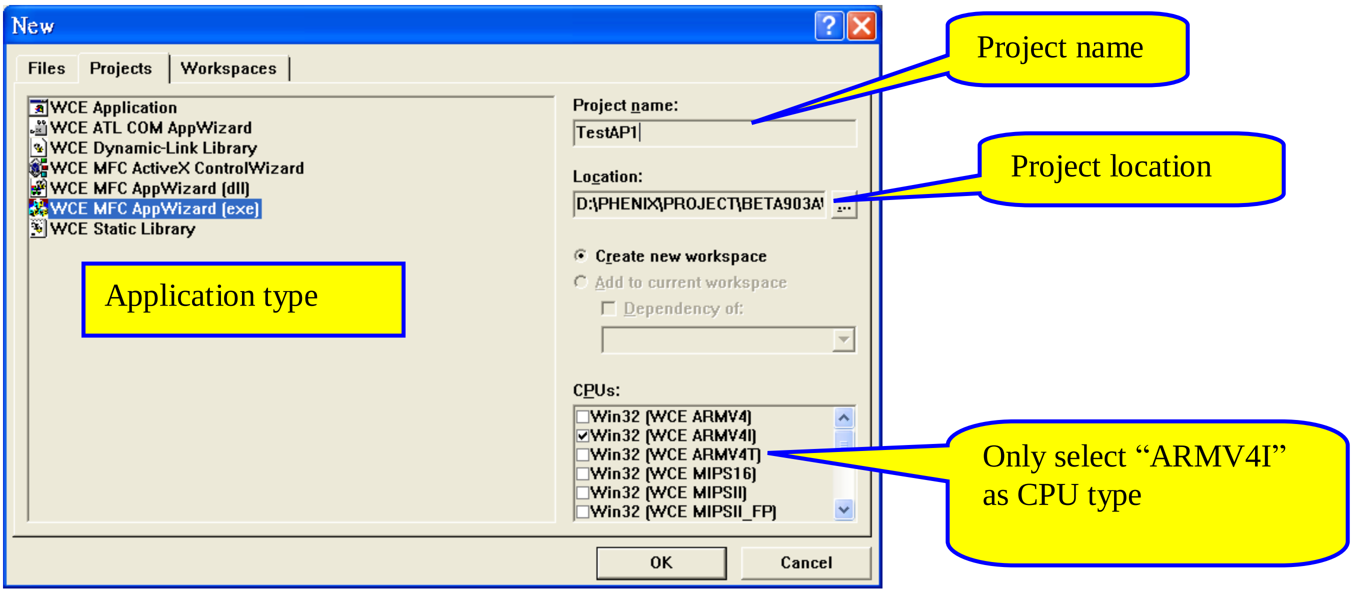
In this section, we will describe how to create a new project in eVC 4.0. An experienced programmer may jump to next section directly. You may create a new project for your application by following steps:

STEP 1: Execute eVC 4.0.

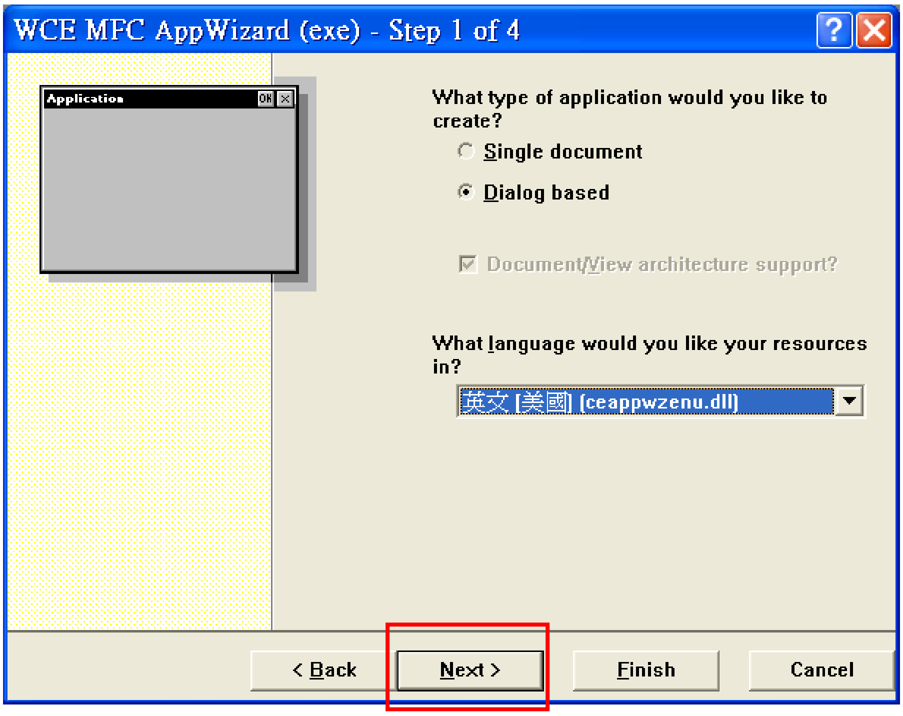
STEP 2: Select "File"-"New…: function



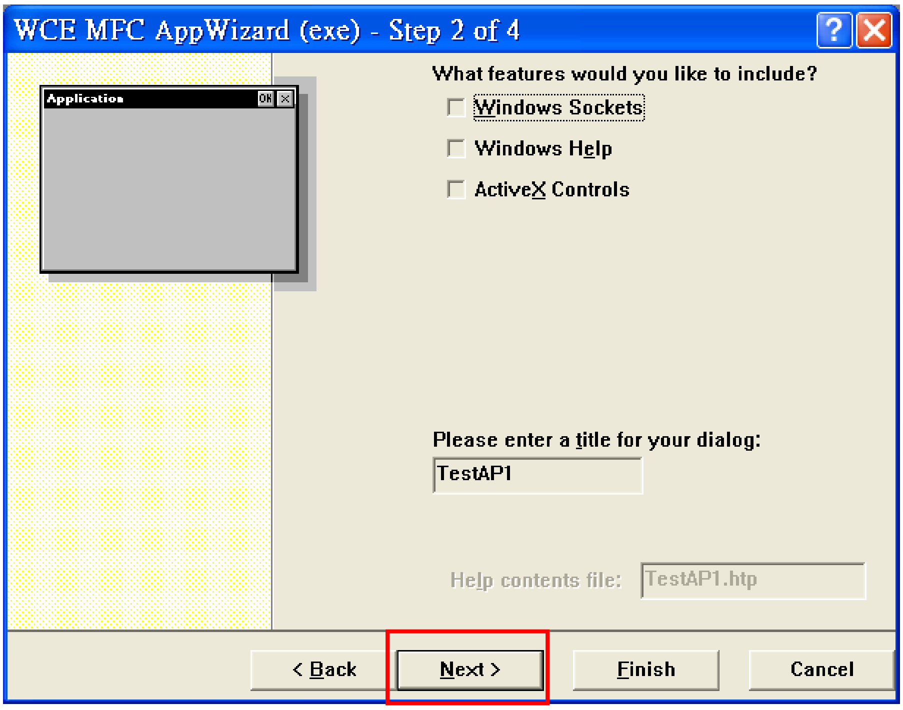
STEP 3: Select your application type, setup the location and name of your project and. Please select "WCE MFC AppWizard(exe)" as application type.



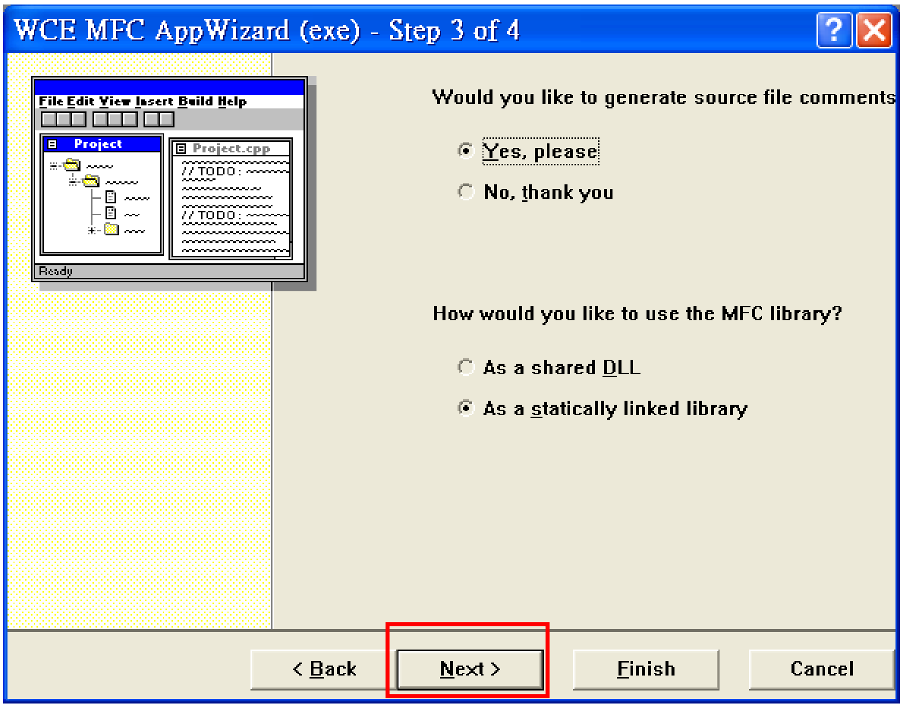
STEP 4: Select "Dialog based" and language setting. Click "Next" button.



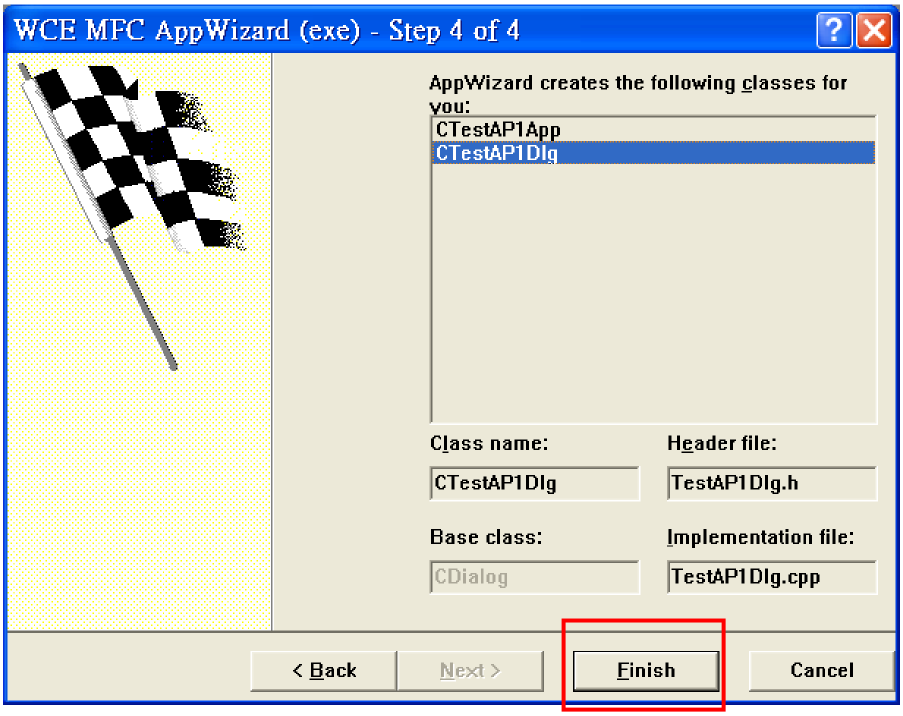
STEP 5: Click "Next" button.



STEP 6: Click "Next" button.



STEP 7: Click "Finish" button.



STEP 8: Now you can add your codes into this new project.

You can get more development information from below MSDN website.
http://msdn.microsoft.com/en-us/library/bb847963.aspx

3.3 Serial Port Function

3.3.1 Overview

There are 4 serial ports in BEGA220A. Below table lists the function of each serial port:

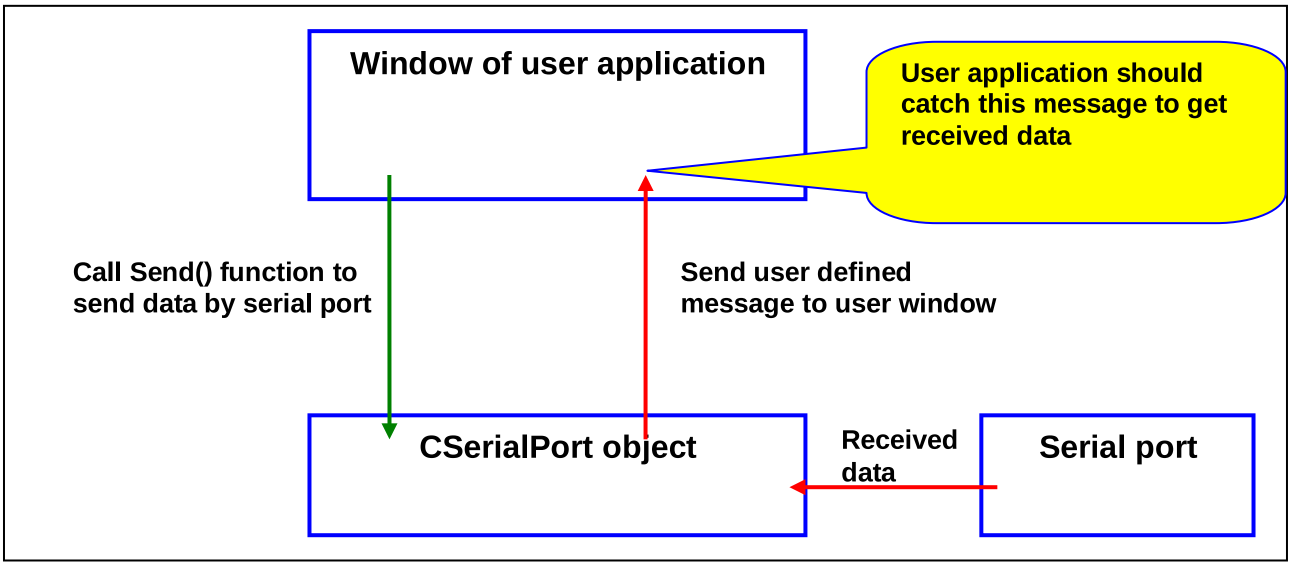
Name	Function	Comment
COM1:	RS-422 or RS485 port	Option
COM3:	Used by application program.	2 wire RS-232
COM4:	Used by application program.	2 wire RS-232
Debug port	Internal used.	May not open by application program.

3.3.2 Serial Port Control-CSerialPort class

Bolymin provided a class, CSerialPort, which implements basic control logic for serial port. Application may use this class by adding **"CSerialPort.cpp"** and **"CSerialPort .h"** into project. Customer may modify the source code of class CSerialPort to expand the serial port functions.

3.3.2.1 Basic concept of class CSerialPort

The object of class CSerialPort will handle all data transfer and receive of opened serial port. Once there is any data is received by the opened serial port, CSerialPort object will send a user defined message to user defined window which should be main window of application program. Below picture shows the flow:



3.3.2.2 Member function of class CSerialPort

CSerialPort Function: Constructor function of calss CSerialPort.

Syntax	CSerialPort();
Parameters	None
Return value	None

Open Function: Open a serial port.

Syntax	BOOL Open(LPCTSTR port, int baud_rate, int data_bit int stop_bit int parity);
Parameters	port Name of serial port listed in the table of section 3.3.1. baud_rate Baud rate, ex: 9600. data_bit Data_bit, 7 ~ 8 stop_bit Stop bit , ONESTOPBIT, ONE5STOPBITS or TWOSTOPBITS. parity Parity , NOPARITY, ODDPARITY, EVENPARITY.
Return value	TRUE: Open serial port successfully FALSE: .Open serial port fail.

Send Function: Send specified data by this serial port.

Syntax	BOOL Send(LPCVOID buf_ptr, DWORD data_len);
Parameters	buf_ptr Memory pointer of data will be sent. data_len Length of data will be sent. (UNIT: byte)
Return value	TRUE: Send data successful. FALSE: Send data fail.

SetCommMsg Function: CSerialPort object will send a receive message to specified window. User need to call this function to set the receive message value and the window that will receive message.

Syntax	void SetCommMsg(HWND win_handle, UINT receive_msg);
Parameters	win_handle Handle of the window that will receive message. receive_msg User defined message value.
Return value	None

Close Function: Close current serial port.

Syntax	BOOL Close ( );
Parameters	None
Return value	TRUE: Close serial port successfully. FALSE: Cloas serial port fail.

3.3.2.3 How to catch the receive message

Please follow below steps to catch the receive message.

STEP 1: Define a receive message in your code as below:
const UINT WM_CMD_OK = WM_USER+1;

STEP 2: Declare a message processing function in the window that will process receive message.

```
// Generated message map functions
//{{AFX_MSG(CSerialPortDlg)
virtual BOOL OnInitDialog();
afx_msg void OnOpenCom();
afx_msg void OnCloseCom();
afx_msg void OnSend();
afx_msg void OnClearSend();
afx_msg void OnClearRec();
afx_msg void OnDestroy();
afx_msg void OnCmdTest();
//}}AFX_MSG
afx_msg LRESULT OnCommRecv(WPARAM wParam, LPARAM lParam);
DECLARE_MESSAGE_MAP()
```

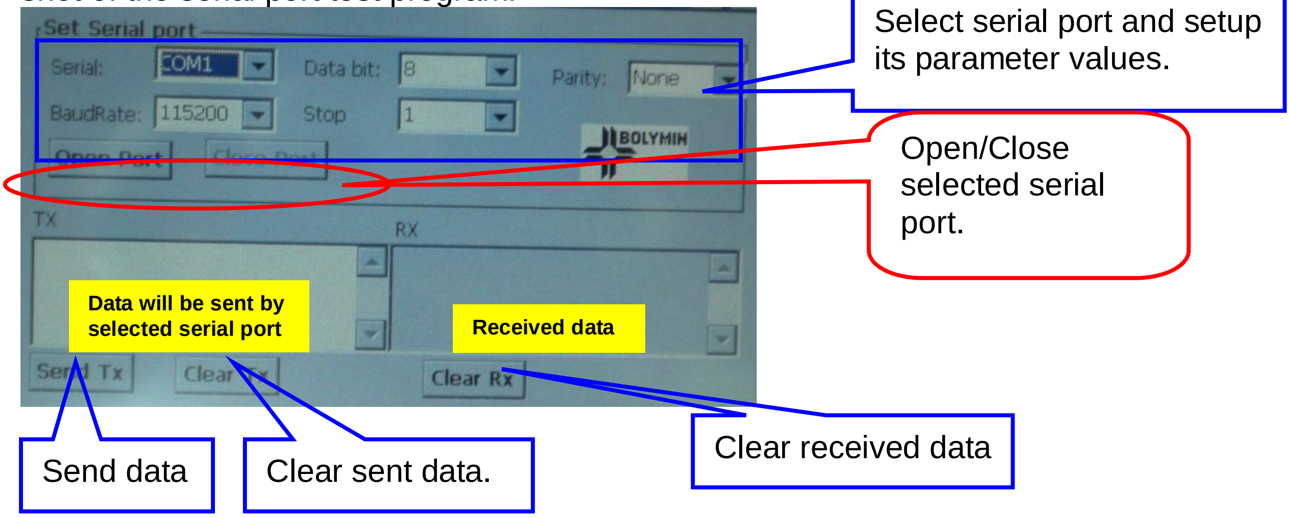
STEP 3: Create message mapping.

```
BEGIN_MESSAGE_MAP(CSerialPortDlg, CDialog)
    //{{AFX_MSG_MAP(CSerialPortDlg)
    ON_BN_CLICKED(IDC_OPEN_COM, OnOpenCom)
    ON_BN_CLICKED(IDC_CLOSE_COM, OnCloseCom)
    ON_BN_CLICKED(IDC_SEND, OnSend)
    ON_BN_CLICKED(IDC_CLEAR_SEND, OnClearSend)
    ON_BN_CLICKED(IDC_CLEAR_REC, OnClearRec)
    ON_BN_CLICKED(IDC_CMD_TEST, OnCmdTest)
    ON_WM_DESTROY()
    //}}AFX_MSG_MAP
    ON_MESSAGE(WM_CMD_OK, OnCommRecv)
END_MESSAGE_MAP()
```

STEP 4: Implement the receive message processing function.

3.3.3 Example Code

Bolymin provide a test application and its source code for example. Below picture is the screen shot of the serial port test program:



Below are the major source codes of the test program:

File: SerialPortDlg.cpp

```
/////////////////////////////////////////////////////////////////////////////
// CSerialPortDlg dialog
/////////////////////////////////////////////////////////////////////////////

const UINT WM_CMD_OK = WM_USER+1;

BEGIN_MESSAGE_MAP(CSerialPortDlg, CDialog)
	//{{AFX_MSG_MAP(CSerialPortDlg)
	ON_BN_CLICKED(IDC_OPEN_COM, OnOpenCom)
	ON_BN_CLICKED(IDC_CLOSE_COM, OnCloseCom)
	ON_BN_CLICKED(IDC_SEND, OnSend)
	ON_BN_CLICKED(IDC_CLEAR_SEND, OnClearSend)
	ON_BN_CLICKED(IDC_CLEAR_REC, OnClearRec)
	ON_WM_DESTROY()
	//}}AFX_MSG_MAP
	ON_MESSAGE(WM_CMD_OK, OnCommRecv)

END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CSerialPortDlg message handlers
/////////////////////////////////////////////////////////////////////////////
```

Define a receive message

Map the receive message processing function to user defined message.

```
BOOL CSerialPortDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	CenterWindow(GetDesktopWindow());	// center to the hpc screen
	m_ComboBaud.SetCurSel(5);		/* Define BaudRate: 115200 */
	m_ComboData.SetCurSel(1);		/* Define data bit: 8 bit */
	m_ComboParity.SetCurSel(0);		/* Define parity: none */
	m_ComboPort.SetCurSel(0);		/* Define searial port: COM1 */
	m_ComboStop.SetCurSel(0);		/* Define stop bit: 1bit */

	m_ButClose.EnableWindow(FALSE);		/* "Close"Button is disable*/
	m_strRecDisp = _T("");
	m_cSendBuffer = new char[60];
	UpdateData(FALSE);

	m_pSerialPort = new CSerialPort();
	m_pSerialPort->SetCommMsg(m_hWnd, WM_CMD_OK);
	return TRUE;
}
```

Create a CSerialPort object and set current window as the window which will process received data.

```
/**********************************************************************************************
Implement function used to process receive data from serial port
**********************************************************************************************/
LRESULT CSerialPortDlg::OnCommRecv(WPARAM wParam, LPARAM lParam)
{
	CString tmp;
	char *buf;
	DWORD buflen;

	buf = (char *)wParam;		// memory pointer of received data
	buflen = (DWORD)lParam;		// received data length

	CEdit *pRecvStrEdit = (CEdit*)GetDlgItem(IDC_REC_DISP);

	for (int i = 0; i < buflen; i++, buf++)
```

```
    {
        tmp.Format(_T("%c"), *buf);
        m_strRecDisp += tmp;
    }

    pRecvStrEdit->SetWindowText(m_strRecDisp);      /* Show */
    return 0;
}

// Initial user interface
const CString PorTbl[4] = {_T("COM1:"),_T("COM3:"),_T("COM4:"),_T("COM6:")};
const DWORD BaudTbl[6] = {4800, 9600, 19200, 38400, 57600,115200};
const DWORD DataBitTbl[2] = {7, 8};
const BYTE   StopBitTbl[3] = {ONESTOPBIT, ONE5STOPBITS, TWOSTOPBITS};
const BYTE   ParityTbl[4] = {NOPARITY, ODDPARITY, EVENPARITY, MARKPARITY};

/***************************************************************************************
Function for “OPEN” button used to open selected serial port.
***************************************************************************************/
void CSerialPortDlg::OnOpenCom()
{
    UpdateData(TRUE);

    CString strPort = PorTbl[m_ComboPort.GetCurSel()];
    DWORD baud   = BaudTbl[m_ComboBaud.GetCurSel()];
    DWORD databit = DataBitTbl[m_ComboData.GetCurSel()];
    BYTE stopbit     = StopBitTbl[m_ComboStop.GetCurSel()];
    BYTE parity      = ParityTbl[m_ComboParity.GetCurSel()];

    BOOL ret = m_pSerialPort->Open(strPort, baud, databit, stopbit, parity);
    if (ret == FALSE)
    {
        MessageBox(_T("Open ") + strPort + _T(" Fail!"));
        return;
    }

    m_ButOpen.EnableWindow(FALSE);        /* Disable "open" button   */
    m_ButClose.EnableWindow(TRUE);        /* Enable "close" button   */
    MessageBox(_T("Open ") + strPort + _T(" is OK!"));
}
```

Open selected serial port by specified parameter values.

```
/*****************************************************************************************
Function for "CLOSE" button used to close current serial port.
*****************************************************************************************/
void CSerialPortDlg::OnCloseCom()
{
        m_pSerialPort->Close();

        m_ButOpen.EnableWindow(TRUE);           /* Enable "Open" button */
        m_ButClose.EnableWindow(FALSE);         /* Disable "close" button    */
}
```

Close current serial port.

```
/*****************************************************************************************
Function for "SEND" button used to send data by serial port.
*****************************************************************************************/
void CSerialPortDlg::OnSend()
{
        UpdateData(TRUE);

        int len = m_strSendEdit.GetLength();

        for(int i = 0; i < len;i++)
                m_cSendBuffer[i] = (char)m_strSendEdit.GetAt(i);;

        BOOL status = m_pSerialPort->Send(m_cSendBuffer, len);
        if (!status)
                MessageBox(_T("Can't write string to COM"),_T("Error"),MB_OK);
}
```

Send data by current serial port.

```
/*****************************************************************************************
Destory function of serial port test dialog
*****************************************************************************************/
void CSerialPortDlg::OnDestroy()
{
        CDialog::OnDestroy();

        m_pSerialPort->Close();
        delete m_pSerialPort;
        delete m_cSendBuffer;
}
```

Close current serial port and delete CSerialPort object.

3.4 GPIO Control

3.4.1 How to Control GPIO for BEGA220A

Bolymin provides a DLL file "**pGPIO_220A.dll**",which is already included in your WinCE OS as a hidden file, to control the General Purpose Input and Ouput(GPIO) signal. In BEGA220A, there are 12 user defined GPIO. User may read current value of all GPIO of BEGA220A, change values of GP output signal and set the attribute of GPIO by functions in "**pGPIO_220A.dll**".

User may use GPIO control functions by following procedures:
STEP 1. Add **"pGPIO_220A.h"** into project.

STEP 2. Load "**pGPIO_220A.dll**" by "**Loadlibrary**()" function.

STEP 3. Get the address of control functions by "**GetProcAddress**()" function.

STEP 4. Execute GPIO control functions by the address got at STEP3.

Below is a simple example code to use the GPIO control functions:

```
// variable declaration
HINSTANCE m_hModule;
BOOL (*m_pGetGPInput)(int);
void (*m_pSetGPOutput)(int, BOOL);
BOOL (*m_pIsOutput)(int);
void (*m_pSetIOAttribute)(int, BOOL);

m_hModule=::LoadLibrary(_T("pGPIO_220A.dll"));
m_pGetGPInput = (BOOL (*)(int))::GetProcAddress(m_hModule,_T("GetGPInput"));
m_pSetGPOutput = (void (*)(int, BOOL))::GetProcAddress(m_hModule,_T("SetGPOutput"));
m_pIsOutput = (BOOL (*)(int))::GetProcAddress(m_hModule,_T("IsOutput"));
m_pSetIOAttribute = (void (*)(int, BOOL))::GetProcAddress(m_hModule,_T("SetIOAttribute"));

m_pSetIOAttribute(GIO_KEY1, GA_INPUT);
m_bPOUT1 = m_pGetGPInput(GIO_KEY1);
m_pSetIOAttribute(GIO_KEY1, GA_OUTPUT);
if (m_pIsOutput(GIO_KEY1))
        m_pSetGPOutput(GIO_KEY1, TRUE);
```

Load "**pGPIO_220A.dll**" and get the address of GPIO control functions.

Execute GPIO control functions.

3.4.2 GPIO Control Function for BEGA220A

GetGPInput Function: Get current status of specified GPIO.

Syntax	BOOL GetGPInput (int gpio_index);
Parameters	gpio_index The index of specified GPIO. Refer to section 3.4.3 for the value definition.
Return value	TRUE: Current status of specified GPIO is HIGH. FALSE: Current status of specified GPIO is LOW.

SetGPOutput Function: Set value of specified GP Output.

Syntax	void SetGPOutput (int gpio_index, BOOL value);
Parameters	gpio_index The index of specified GP output. Refer to section 3.4.3 for the value definition. value New value of specified GP output. TRUE: Set specified GP output to HIGH. FALSE: Set specified GP output to LOW.
Return value	None

IsOutput Function: Check if the specified GPIO is output or not..

Syntax	BOOL IsOutput (int gpio_index);
Parameters	gpio_index The index of specified GPIO. Refer to section 3.4.3 for the value definition.
Return value	TRUE: The specified GPIO is output. FALSE: The specified GPIO is input.

SetIOAttribute Function: Set the attribute of the specified GPIO.

Syntax	void SetIOAttribute (int gpio_index, BOOL value);
Parameters	gpio_index The index of specified GPIO. **GIO_KEY1~GIO_KEY12**. value New attribute of the specified GPIO. GA_OUTPUT: Set the specified GPIO as output. GA_INPUT: Set the specified GPIO as input.
Return value	None

3.4.3 Definition of GPIO Index

Class CGPIO_220A support following index values:

GPIO index	Description
GIO_KEY1	User defined general purpose input/output. (KEY1)
GIO_KEY2	User defined general purpose input/output. (KEY2)
GIO_KEY3	User defined general purpose input/output. (KEY3)
GIO_KEY4	User defined general purpose input/output. (KEY4)
GIO_KEY5	User defined general purpose input/output. (KEY5)
GIO_KEY6	User defined general purpose input/output. (KEY6)
GIO_KEY7	User defined general purpose input/output. (KEY7)
GIO_KEY8	User defined general purpose input/output. (KEY8)
GIO_KEY9	User defined general purpose input/output. (KEY9)
GIO_KEY10	User defined general purpose input/output. (KEY10)
GIO_KEY11	User defined general purpose input/output. (KEY11)
GIO_KEY12	User defined general purpose input/output. (KEY12)
GO_BLIGHT_ENABLE	Backlight control. Default value : **HIGH**.
GO_LCD_POWER_ENABLE	LCD power control. Default value : **LOW**.
GO_AMP_SWITCH	Amplifier switch. Default vale: **LOW**.
GO_EN485	Reserved for internal use.

3.5 ADC Converter and Backlight Adjustment

3.5.1 Overview

Bolymin provides a dynamic link library "**CtrlFunc_220A.dll**" which includes all control functions of A/D converter and backlight adjustment.

- For A/D converter, there are 6-ch A/D converters with 12-bit resolution in BEGA220A. User may read the value from selected A/D converter channel by functions provided by Bolymin. Suggested hardware wiring about A/D converter may be found in user manual. Detail description of control functions of A/D converter may be found in section 3.5.2.
- For backlight adjustment, user may get current brightness value or change the brightness of backlight by control functions. The description of control functions of backlight adjustment may be found in section 3.5.3.

User may use control functions about A/D converter and backlight adjustment by following procedures:

STEP 1. Add **"CtrlFunc_220A.h"** into project.
STEP 2. Load "**CtrlFunc_220A.dll**" by "**Loadlibrary**()" function.
STEP 3. Get the address of control functions by "**GetProcAddress**()" function.
STEP 4. Execute control functions about A/D converter by the address got at STEP3.
STEP 5. Call **"FreeLibrary()"** function to free the reference of "**CtrlFunc_220A.dll**" while ending the application program.

Below is a simple example code to use the control functions about A/D converter and backlight adjustment:

```
// variable declaration
HINSTANCE m_hModule;
BOOL (*m_pInitADC)(void);
int (*m_pReadADC)(int);

m_hModule=::LoadLibrary(_T("CtrlFunc_220A.dll"));
m_pInitADC = (BOOL (*)(void))::GetProcAddress(m_hModule,_T("InitADC"));
m_pReadADC = (int (*)(int))::GetProcAddress(m_hModule,_T("ReadADC"));
m_pInitBacklightCtrl = (BOOL (*)(void))::GetProcAddress(m_hModule,_T("InitBacklightCtrl"));
m_pGetBrightness = (int (*)(void))::GetProcAddress(m_hModule,_T("GetBrightness"));
m_pSetBrightness = (int (*)(int))::GetProcAddress(m_hModule,_T("SetBrightness"));

m_pInitADC();
value = m_pReadADC(ADC_CHANNEL0);

m_pInitBacklightCtrl();
brightness = m_pGetBrightness();
m_pSetBrightness(brightness+5);   // Increase the brightness by 5
m_pSetBrightness(0);              // OFF the backlight

FreeLibrary(m_hModule);           // free the reference of "CtrlFunc_220A.dll"
```

Load "**CtrlFunc_220A.dll**" and get the address of control functions.

Execute control functions of A/D converter.

Execute control functions of backlight adjustment.

3.5.2 Control Function of A/D Converter

initADC Function: A/D converter initialization. **User need to call this function before using A/D converter.**

Syntax	BOOL initADC ();
Parameters	None
Return value	TRUE: Initial A/D converter successfully. FALSE: Fail to initial A/D converter.

ReadADC Function: Read A/D converted data from specified A/D converter channel.

Syntax	Int ReadADC(int adc_channel);
Parameters	adc_channel The index of specified A/D converter channel. **ADC_CHANNEL0~ADC_CHANNEL5**
Return value	A/D converted data from specified A/D converter channel.

3.5.3 Function about Backlight Adjustment

InitBacklightCtrl Function: Initial backlight controller. **User need to call this function before adjusting backlight brightness.**

Syntax	BOOL InitBacklightCtrl ();
Parameters	None
Return value	TRUE: Initial backlight controller successfully. FALSE: Fail to initial backlight controller.

GetBrightness Function: Get current brightness value of backlight.

Syntax	int GetBrightness ();
Parameters	None
Return value	Current brightness value of backlight. (0~100)

SetBrightness Function: Set brightness value of backlight.

Syntax	Int SetBrightness(int new_value);
Parameters	new_value New brightness value of backlight. (0~100) **0: Turn OFF the backlight**
Return value	Original brightness value of backlight.

<End of BEGA220A User Manual >